Matematica delle Reti Neurali

Volume IV

Information geometry , Interpretabilità meccanicistica , Causalità , Scaling laws , Privacy differenziale , Audio/Speech

Silvano Salvador

Matematica delle Reti Neurali

Information geometry · Interpretabilità meccanicistica · Causalità · Scaling laws · Privacy differenziale · Audio/Speech · Robotica · Biologia computazionale

Silvano Salvador · Continuazione dei Volumi I–III (Parti 0–178)

Parti 201–246 Fisher-Rao e Natural Gradient Circuits e Sparse Autoencoders Do-calculus e IRM Rényi DP e DP-SGD CTC · RNN-T · HiFi-GAN Diffusion Policy · RT-2 AlphaFold2 completo

Introduzione al Volume IV

Otto aree non coperte nei volumi precedenti, scelte per la densità matematica e la rilevanza pratica.

I tre volumi precedenti hanno costruito un panorama esteso della matematica delle reti neurali — dalle fondamenta (Vol. I), alle architetture moderne e alla teoria (Vol. II), fino a RL, self-supervised learning, teoria statistica, training distribuito e biologia di base (Vol. III). Il presente volume esplora otto aree rimaste scoperte, ognuna con un proprio corpus matematico ricco.

Sezione A (§201–206): La geometria dell'informazione fornisce il linguaggio per capire perché certi ottimizzatori convergono più velocemente, come si propagano i segnali nelle reti profonde e quali soluzioni SGD preferisce implicitamente. Si derivano la metrica di Fisher-Rao, il natural gradient, K-FAC come approssimazione, la mean field theory con le equazioni di propagazione del segnale, e il teorema sull'implicit bias verso la soluzione a margine massimo.

Sezione B (§207–212): L'interpretabilità meccanicistica è la ricerca di algoritmi concreti implementati dai pesi delle reti. Si trattano feature visualization con metodo delle feature parassite, la superposition hypothesis con geometria degli antipodi, gli sparse autoencoders come dizionario di feature monosemantiche, i circuiti (induction head, copy suppression), il causal tracing per la localizzazione del fatto, e il logit lens per la decodifica intermedia delle rappresentazioni.

Sezione C (§213–218): La causalità va oltre la correlazione — si derivano i criteri di d-separazione, il do-calculus di Pearl con la regola del backdoor e del frontdoor, i Structural Causal Models per i controfattuali, e l'Invariant Risk Minimization come metodo di apprendimento causale end-to-end.

Sezione D (§219–224): Le leggi di scaling prevedono le performance dei modelli in funzione di compute, dati e parametri. Si derivano le equazioni di Kaplan e di Chinchilla, si analizza il fenomeno delle abilità emergenti, l'inference-time compute scaling, la curazione dei dati e l'architettura di sistemi di inferenza efficienti come vLLM con PagedAttention.

Sezione E (§225–230): La privacy differenziale fornisce garanzie matematiche rigorose sulla protezione dei dati. Si derivano la definizione (ε, δ)-DP, il meccanismo gaussiano, la composizione, la Rényi DP con il moments accountant, DP-SGD con analisi del noise, gli attacchi di membership inference e l'integrazione con federated learning.

Sezione F (§231–236): La sintesi e il riconoscimento audio richiedono rappresentazioni specifiche (STFT, mel, MFCC), architetture per i vocoder (HiFi-GAN, BigVGAN), algoritmi per l'ASR (CTC con la forward-backward algorithm, RNN-Transducer), il modello multitask di Whisper, e i codec language models (AudioLM, MusicGen).

Sezione G (§237–241): La robotica moderna usa reti neurali per perception, planning e control. Si trattano le diffusion policies con la derivazione dello score matching per l'imitazione, i modelli vision-language-action (RT-1, RT-2), il MPC neurale, SLAM con rappresentazioni neurali e NeRF per la manipolazione 3D.

Sezione H (§242–246): La biologia computazionale è un banco di prova per le architetture più avanzate. Si derivano i protein language models (ESM, ProtTrans), l'Evoformer di AlphaFold2 con l'invariant point attention, i modelli generativi per drug discovery (REINFORCE per grafici molecolari, flow matching per molecole), i DNA language models e le GNN geometriche per molecole (SchNet, DimeNet, PaiNN).

Riferimento ai Volumi precedenti: la notazione segue la Sezione 0 del Volume I. Rimandi espliciti indicano §N del Vol. I, Vol. II o Vol. III.

Varietà Statistiche e Metrica di Fisher-Rao

201.1 Lo Spazio delle Distribuzioni come Varietà Riemanniana

La geometria dell'informazione (Amari, 1985) interpreta famiglie parametriche di distribuzioni $\{p(\mathbf{x};\boldsymbol{\theta})\}_{\boldsymbol{\theta}\in\Theta}$ come **varietà statistiche** — spazi curvi con una geometria intrinseca. Il parametro $\boldsymbol{\theta} \in \mathbb{R}^n$ funge da sistema di coordinate sulla varietà.

La metrica naturale su questa varietà è la **metrica di Fisher-Rao**. Il tensore metrico $\mathbf{G}(\boldsymbol{\theta})$ con elementi:

$$G_{ij}(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{x};\boldsymbol{\theta})}\left[\frac{\partial \log p(\mathbf{x};\boldsymbol{\theta})}{\partial \theta_i} \cdot \frac{\partial \log p(\mathbf{x};\boldsymbol{\theta})}{\partial \theta_j}\right] = -\mathbb{E}_p\left[\frac{\partial^2 \log p(\mathbf{x};\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j}\right]$$

Le due espressioni sono equivalenti per la identity di Fisher: $\mathbb{E}_p[\partial_\theta \log p] = 0$ (la score function ha media zero). Il tensore $\mathbf{G}(\boldsymbol{\theta})$ è la **Fisher Information Matrix (FIM)** — misura quanto la distribuzione cambia al variare di $\boldsymbol{\theta}$.

Proprietà geometriche della FIM

La FIM definisce una **distanza infinitesima** tra distribuzioni vicine $p(\mathbf{x};\boldsymbol{\theta})$ e $p(\mathbf{x};\boldsymbol{\theta}+d\boldsymbol{\theta})$:

$$ds^2 = d\boldsymbol{\theta}^T \mathbf{G}(\boldsymbol{\theta})\, d\boldsymbol{\theta} = \sum_{i,j} G_{ij}(\boldsymbol{\theta})\, d\theta_i\, d\theta_j$$

Questa è proporzionale alla divergenza di KL al secondo ordine: $D_{\mathrm{KL}}(p(\cdot;\boldsymbol{\theta})\|p(\cdot;\boldsymbol{\theta}+d\boldsymbol{\theta})) = \frac{1}{2} d\boldsymbol{\theta}^T \mathbf{G}(\boldsymbol{\theta})\, d\boldsymbol{\theta} + O(|d\boldsymbol{\theta}|^3)$. La metrica di Fisher-Rao è la *unica* metrica invariante rispetto a reparametrizzazioni sufficienti (teorema di Čencov, 1982).

Esempi espliciti

Famiglia	**Parametri $\boldsymbol{\theta}$**	**FIM $\mathbf{G}(\boldsymbol{\theta})$**
Gaussiana $\mathcal{N}(\mu, \sigma^2)$	(μ, σ^2)	$\mathrm{diag}(1/\sigma^2,\, 1/(2\sigma^4))$
Bernoulli $p^k(1-p)^{1-k}$	p	$1/(p(1-p))$
Softmax $e^{\theta_k}/\sum e^{\theta_j}$	$\boldsymbol{\theta} \in \mathbb{R}^K$	$\mathrm{diag}(\mathbf{p}) - \mathbf{p}\mathbf{p}^T$ (singolare)
Gaussiana multivar. $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathbf{G}_{\mu\mu} = \boldsymbol{\Sigma}^{-1}$, $\mathbf{G}_{\Sigma\Sigma,ijkl} = \frac{1}{2}\Sigma^{-1}_{ik}\Sigma^{-1}_{jl}$

Per la softmax, la FIM è singolare perché la parametrizzazione è ridondante ($\boldsymbol{\theta}$ e $\boldsymbol{\theta} + c\mathbf{1}$ definiscono la stessa distribuzione) — riflette la struttura di iperpiano della varietà del simplesso.

201.2 Connessioni Duale: e-connection e m-connection

La geometria dell'informazione introduce due connessioni affini duali sulla varietà statistica, parametrizzate da $\alpha \in \mathbb{R}$. Le connessioni α-Christoffel:

$$\Gamma^{(\alpha)}_{ij,k} = \mathbb{E}_p\left[\left(\partial_i\partial_j\ell + \frac{1-\alpha}{2}\partial_i\ell\,\partial_j\ell\right)\partial_k\ell\right]$$

dove $\ell = \log p(\mathbf{x}; \boldsymbol{\theta})$ è il log-likelihood. I casi notevoli: $\alpha = +1$ è la e-connection (esponenziale, usata per famiglie esponenziali); $\alpha = -1$ è la m-connection (mixture, duale); $\alpha = 0$ è la connessione di Levi-Civita (compatibile con la metrica).

La **dualità**: geodetiche della e-connection minimizzano $D_{\mathrm{KL}}(p\|q)$; geodetiche della m-connection minimizzano $D_{\mathrm{KL}}(q\|p)$. Questa asimmetria riflette quella della KL divergence.

Natural Gradient e K-FAC

202.1 Il Natural Gradient di Amari

Il gradiente euclideo $\nabla_\theta \mathcal{L}$ dipende dalla parametrizzazione — cambiando coordinate, la direzione di discesa cambia. Il **natural gradient** è invariante rispetto alla reparametrizzazione: è il gradiente nella geometria della varietà statistica definita dalla FIM.

Formalmente, si cerca la perturbazione $d\boldsymbol{\theta}$ che minimizza la loss al primo ordine, soggetta al vincolo che la distribuzione cambi di una quantità fissa: $\min_{d\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} + d\boldsymbol{\theta})$ s.t. $D_{\mathrm{KL}}(p(\cdot; \boldsymbol{\theta} + d\boldsymbol{\theta})\|p(\cdot; \boldsymbol{\theta})) = \varepsilon$.

$\min_{d\boldsymbol{\theta}} \nabla\mathcal{L}^T d\boldsymbol{\theta}$ s.t. $\frac{1}{2} d\boldsymbol{\theta}^T \mathbf{G} d\boldsymbol{\theta} = \varepsilon$ Linearizzazione della loss e approssimazione quadratica del vincolo KL

Lagrangiano: $\nabla\mathcal{L}^T d\boldsymbol{\theta} - \lambda(\frac{1}{2} d\boldsymbol{\theta}^T \mathbf{G} d\boldsymbol{\theta} - \varepsilon)$ Moltiplicatore di Lagrange

Condizione stazionaria: $\nabla\mathcal{L} = \lambda \mathbf{G} d\boldsymbol{\theta}$, quindi $d\boldsymbol{\theta} = \frac{1}{\lambda}\mathbf{G}^{-1}\nabla\mathcal{L}$ Il vincolo determina λ

$$\tilde{\nabla}\mathcal{L}(\boldsymbol{\theta}) = \mathbf{G}(\boldsymbol{\theta})^{-1}\nabla_\theta \mathcal{L}(\boldsymbol{\theta}) \qquad \text{(Natural Gradient)}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\, \mathbf{G}(\boldsymbol{\theta}_t)^{-1}\nabla\mathcal{L}(\boldsymbol{\theta}_t)$$

Il natural gradient è invariante: sotto la reparametrizzazione $\phi = f(\theta)$, $\tilde{\nabla}_\phi \mathcal{L} = \mathbf{G}_\phi^{-1}\nabla_\phi \mathcal{L}$ produce la stessa direzione di aggiornamento nello spazio della distribuzione.

202.2 K-FAC: Kronecker-Factored Approximate Curvature

Il problema principale del natural gradient: calcolare e invertire $\mathbf{G}(\boldsymbol{\theta}) \in \mathbb{R}^{|\theta|\times|\theta|}$ ha costo $O(|\theta|^3)$ — impraticabile per reti moderne. K-FAC (Martens e Grosse, 2015) approssima la FIM come prodotto di Kronecker per ogni layer.

Per un layer lineare con input $\mathbf{a} \in \mathbb{R}^{d_{\text{in}}}$ e output $\mathbf{s} = \mathbf{W}\mathbf{a}$ (pre-attivazione), il gradiente dei pesi è $\nabla_{\mathbf{W}} \ell = \mathbf{g}\mathbf{a}^T$ dove $\mathbf{g} = \nabla_{\mathbf{s}} \ell$ è il backpropagated gradient. Il blocco della FIM per $\mathbf{W}$:

$$\mathbf{G}_W = \mathbb{E}[(\mathbf{g}\mathbf{a}^T) \otimes (\mathbf{g}\mathbf{a}^T)] = \mathbb{E}[\mathbf{g}\mathbf{g}^T \otimes \mathbf{a}\mathbf{a}^T]$$

K-FAC approssima fattorizzando l'aspettativa del prodotto come prodotto delle aspettative — indipendenza tra attivazioni e gradienti retropropagati:

$$\hat{\mathbf{G}}_W \approx \underbrace{\mathbb{E}[\mathbf{g}\mathbf{g}^T]}_{\mathbf{\Omega} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}} \otimes \underbrace{\mathbb{E}[\mathbf{a}\mathbf{a}^T]}_{\mathbf{\Gamma} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}}$$

L'inversione sfrutta la proprietà del prodotto di Kronecker: $(\mathbf{\Omega} \otimes \mathbf{\Gamma})^{-1} = \mathbf{\Omega}^{-1} \otimes \mathbf{\Gamma}^{-1}$, riducendo il costo da $O((d_{\text{in}} d_{\text{out}})^3)$ a $O(d_{\text{in}}^3 + d_{\text{out}}^3)$. L'aggiornamento K-FAC:

$$\Delta \mathbf{W} = -\eta \, \mathbf{\Omega}^{-1} \nabla_{\mathbf{W}} \mathcal{L} \, \mathbf{\Gamma}^{-1}$$

K-FAC converge più velocemente di SGD/Adam in meno step, ma ogni step è più costoso. In pratica si aggiornano $\mathbf{\Omega}$ e $\mathbf{\Gamma}$ ogni T_{stat} step e si rinverte ogni T_{inv} step — ammortizzando il costo.

Geometria Riemanniana nell'Ottimizzazione

203.1 Retrazione e Ottimizzazione su Varietà

Molti problemi di ML hanno vincoli geometrici naturali: pesi ortogonali risiedono sulla varietà di Stiefel, le distribuzioni di probabilità sul simplesso, gli embedding sulla sfera unitaria. Proiettare gradient euclidei su questi vincoli ad ogni step è costoso e introduce discontinuità agli angoli del vincolo. L'**ottimizzazione Riemanniana** risolve il problema alla radice, generalizzando il gradient descent a varietà differenziali: invece di muoversi in linea retta nello spazio euclideo e poi proiettare, ci si muove lungo geodetiche sulla varietà usando la metrica intrinseca.

Su una varietà $\mathcal{M}$, il **Riemannian gradient** $\text{grad}_{\mathcal{M}} f(\mathbf{x})$ è la proiezione del gradiente euclideo sullo spazio tangente $T_{\mathbf{x}}\mathcal{M}$:

$$\text{grad}_{\mathcal{M}} f(\mathbf{x}) = \Pi_{T_{\mathbf{x}}\mathcal{M}}(\nabla f(\mathbf{x}))$$

L'aggiornamento invece di $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \nabla f$ usa una **retrazione** $R_{\mathbf{x}} : T_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$ che riporta il punto aggiornato sulla varietà:

$$\mathbf{x}_{t+1} = R_{\mathbf{x}_t}(-\eta \, \text{grad}_{\mathcal{M}} f(\mathbf{x}_t))$$

Esempi di varietà e loro retrazioni

Varietà	**Spazio tangente $T_{\mathbf{x}}\mathcal{M}$**	**Proiezione Π**	**Retraz**
Sfera S^{n-1}: $\|\mathbf{x}\| = 1$	$\{\mathbf{v} : \mathbf{x}^T\mathbf{v} = 0\}$	$\mathbf{v} - (\mathbf{x}^T\mathbf{v})\mathbf{x}$	$(\mathbf{x}+\mathbf{v})/$
Simplesso Δ^{n-1}: $\sum x_i = 1, x_i \geq 0$	$\{\mathbf{v} : \sum v_i = 0\}$	$\mathbf{v} - \frac{\sum v_i}{n}\mathbf{1}$	softmax
Stiefel $\text{St}(n,p)$: $\mathbf{X}^T\mathbf{X} = \mathbf{I}_p$	$\{\mathbf{Z} : \mathbf{X}^T\mathbf{Z} + \mathbf{Z}^T\mathbf{X} = 0\}$	$\mathbf{Z} - \mathbf{X}\text{sym}(\mathbf{X}^T\mathbf{Z})$	QR dec
Grassmannian $\text{Gr}(n,p)$: sottospazi di $\mathbb{R}^n$	Matrici antisimm. rispetto a $\mathbf{X}$	$(\mathbf{I} - \mathbf{X}\mathbf{X}^T)\mathbf{Z}$	SVD-ba

L'ottimizzazione su Stiefel (pesi ortogonali) garantisce stabilità del gradiente — la norma degli autovalori rimane 1. La varietà Grassmanniana è usata nell'attention multi-head per forzare testa diverse a catturare sottospazi ortogonali.

Mean Field Theory: Propagazione del Segnale nelle Reti Profonde

204.1 Teoria del Campo Medio per Reti Fully-Connected

La mean field theory (Poole et al., 2016; Schoenholz et al., 2017) analizza la propagazione delle attivazioni e dei gradienti in reti profonde MLP all'inizializzazione casuale. I pesi $W_{ij}^l \sim \mathcal{N}(0, \sigma_w^2/N)$ e i bias $b_i^l \sim \mathcal{N}(0, \sigma_b^2)$.

Per $N \to \infty$ (larghezza infinita), il pre-activation $h_i^l = \sum_j W_{ij}^l \phi(h_j^{l-1}) + b_i^l$ converge a un processo gaussiano per il Teorema Centrale del Limite. La varianza delle pre-activazioni si propaga seguendo una **mappa di campo medio**:

$$q^l = \sigma_w^2 \mathbb{E}_{z\sim\mathcal{N}(0,q^{l-1})}[\phi(z)^2] + \sigma_b^2 \equiv V(\sigma_w, \sigma_b, q^{l-1})$$

Il punto fisso $q^* = V(\sigma_w, \sigma_b, q^*)$ descrive la varianza delle attivazioni a convergenza. Per la funzione ReLU $\phi(z) = \max(0, z)$:

$$q^l = \frac{\sigma_w^2}{2} q^{l-1} + \sigma_b^2$$

$$q^* = \frac{\sigma_b^2}{1 - \sigma_w^2/2} \quad \text{se } \sigma_w^2 < 2$$

204.2 Mappa della Correlazione e Transizione di Fase

Per due input $\mathbf{x}^a$ e $\mathbf{x}^b$, la loro correlazione $c^l = q_{ab}^l / \sqrt{q_{aa}^l q_{bb}^l}$ evolve secondo una seconda mappa:

$$c^l = C(\sigma_w, \sigma_b, c^{l-1}) = \frac{\sigma_w^2 \mathbb{E}_{(z^a, z^b) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^{l-1})}[\phi(z^a)\phi(z^b)]}{q^*}$$

dove $\mathbf{Q}^{l-1} = q^* \begin{pmatrix} 1 & c^{l-1} \\ c^{l-1} & 1 \end{pmatrix}$. La stabilità del punto fisso $c^* = 1$ determina il regime:

Transizione Ordine-Caos

La derivata $\chi_1 = \partial C / \partial c|_{c=1} = \sigma_w^2 \mathbb{E}[\phi'(z)^2]$ determina il regime:

- $\chi_1 < 1$: regime **ordinato** — input diversi producono attivazioni correlate, i gradienti si annullano in profondità (*vanishing gradients*)
- $\chi_1 > 1$: regime **caotico** — input simili producono attivazioni incorrelate, i gradienti esplodono (*exploding gradients*)
- $\chi_1 = 1$: **Edge of Chaos** — massima trasmissione dell'informazione, gradienti stabili

Per ReLU: $\chi_1 = \sigma_w^2/2$. L'**inizializzazione He** $\sigma_w^2 = 2/N$ corrisponde esattamente a $\chi_1 = 1$ — spiega perché questa inizializzazione funziona: porta la rete all'edge of chaos.

Implicit Bias di Gradient Descent

205.1 Verso Soluzioni a Norma Minima

In problemi overparametrizzati, esistono infinite soluzioni con training error zero. SGD non sceglie a caso — converge verso la soluzione con proprietà specifiche. Per regressione lineare

$\mathbf{y} = \mathbf{X}\boldsymbol{\theta}$ con $\mathbf{X} \in \mathbb{R}^{n \times d}$, $d > n$:

Teorema (Implicit Bias di GD per regressione lineare)

Partendo da $\boldsymbol{\theta}_0 = \mathbf{0}$ e con learning rate $\eta < 2/\lambda_{\max}(\mathbf{X}^T\mathbf{X})$, gradient descent converge a $\hat{\boldsymbol{\theta}} = \mathbf{X}^+\mathbf{y}$ — la soluzione a norma minima ($\|\hat{\boldsymbol{\theta}}\|_2$ minima tra tutte le soluzioni interpolanti).

Dimostrazione: GD mantiene $\boldsymbol{\theta}_t \in \text{RowSpace}(\mathbf{X})$ per ogni t (perché $\nabla\mathcal{L} = \mathbf{X}^T(\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \in \text{RowSpace}(\mathbf{X})$ e $\boldsymbol{\theta}_0 = 0$). La soluzione a norma minima è $\mathbf{X}^+\mathbf{y}$ — la componente nell'orthogonal complement del row space è zero. Poiché GD converge all'unica soluzione in $\text{RowSpace}(\mathbf{X})$, converge a $\mathbf{X}^+\mathbf{y}$.

205.2 Verso Soluzioni a Margine Massimo per Classificatori

Per classificazione binaria con dati linearmente separabili, gradient descent su loss esponenziali o logistiche converge verso il **max-margin classifier** (SVM):

Teorema (Soudry et al., 2018)

Per classificazione lineare con loss logistica su dati linearmente separabili, $\boldsymbol{\theta}_t/\|\boldsymbol{\theta}_t\|_2 \to \hat{\boldsymbol{\theta}}_{\text{SVM}}$ (normalizzato) dove $\hat{\boldsymbol{\theta}}_{\text{SVM}}$ è la soluzione del hard-margin SVM. La convergenza è in direzione, non in norma — i pesi divergono a velocità $O(\log t)$.

Questo spiega il fenomeno del **late-stage overfitting**: anche dopo molte epoche su training loss prossima a zero, la direzione del classificatore continua a cambiare verso il max-margin — migliorando o peggiorando la generalizzazione a seconda della geometria dei dati.

205.3 Implicit Bias dei Reti Profonde: Soluzioni Sparse

Per reti profonde lineari $f(\mathbf{x}) = \mathbf{W}_L \cdots \mathbf{W}_1\mathbf{x}$, il prodotto $\mathbf{W} = \mathbf{W}_L \cdots \mathbf{W}_1$ ha un implicit bias verso soluzioni con **norma nucleare piccola** (rango basso):

$$\|\mathbf{W}\|_* = \sum_i \sigma_i(\mathbf{W}) \qquad \text{(norma nucleare = somma valori singolari)}$$

Per profondità $L = 2$: GD con inizializzazione piccola converge verso $\hat{\mathbf{W}} = \arg\min_{\mathbf{W}} \|\mathbf{W}\|_*$ soggetto a $\mathbf{X}\mathbf{W}^T = \mathbf{Y}$ — esattamente il problema di matrix completion / low-rank regularization. La profondità crea un'implicit regularizzazione verso il rango basso senza penalità esplicita.

Mode Connectivity e Geometria del Loss Landscape

206.1 Linear Mode Connectivity

Due soluzioni $\boldsymbol{\theta}_A$ e $\boldsymbol{\theta}_B$ (stessa rete, diverse run) sono **linearmente mode-connected** se il percorso lineare $\boldsymbol{\theta}(\lambda) = (1-\lambda)\boldsymbol{\theta}_A + \lambda\boldsymbol{\theta}_B$ ha barrier bassa: $\max_{\lambda \in [0,1]} \mathcal{L}(\boldsymbol{\theta}(\lambda)) - \min(\mathcal{L}_A, \mathcal{L}_B) \approx 0$.

Frankle et al. (2020) dimostrano che le soluzioni trovate da SGD con lo *stesso* punto di partenza random (ma shuffle diverso dei dati) sono linearmente mode-connected. Con punti di partenza diversi, sono connesse tramite percorsi curvi a bassa barriera ma non lineari.

Permuatation Symmetry e Alignment

Due soluzioni $\boldsymbol{\theta}_A$ e $\boldsymbol{\theta}_B$ non sono mai linearmente connesse se i neuroni sono permutati — bisogna prima **allineare** i neuroni. Il problema di allineamento: trovare la permutazione π^* dei neuroni di ogni layer tale che $\boldsymbol{\theta}_A \approx \boldsymbol{\theta}_B^{\pi}$ (con $\boldsymbol{\theta}^{\pi}$ la rete con neuroni permutati):

$$\pi^* = \arg\min_{\pi} \|\boldsymbol{\theta}_A - \boldsymbol{\theta}_B^{\pi}\|_F^2$$

Dopo allineamento ottimale (risolto con Hungarian algorithm o gradient-based), le soluzioni sono quasi sempre linearmente mode-connected — la **loss barrier è quasi zero** (Ainsworth et al., 2023). Questo implica che lo spazio dei minimi è molto connesso — SGD trova soluzioni tutte "vicine" modulo permutazione.

206.2 Sharpness e SAM — Connessione con Generalizzazione

La sharpness di un minimo misura la curvatura del loss attorno ad esso. La **flatness** (bassa sharpness) è correlata con la buona generalizzazione (Hochreiter e Schmidhuber, 1997; Keskar et al., 2017). Formalmente:

$$\text{Sharpness}(\boldsymbol{\theta}) = \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) - \mathcal{L}(\boldsymbol{\theta})$$

SAM (§80, Vol. II) minimizza esplicitamente la sharpness. La spiegazione geometrica: un minimo piatto è robusto a piccole perturbazioni dei pesi — che possono provenire dalla discretizzazione numerica (quantizzazione), dal rumore del mini-batch SGD, o dalla distribuzione diversa dei dati di test. Un minimo aguzzo può avere training loss minima ma essere circondato da regioni ad alta loss — fragile alla shift distribuzionale.

Feature Visualization e Activation Maximization

207.1 Activation Maximization: Trovare l'Input che Attiva un Neurone

L'**interpretabilità meccanicistica** mira a comprendere i meccanismi computazionali interni alle reti neurali, a livello di neuroni e circuiti. Il suo strumento più immediato è l'**activation maximization**: invece di passare immagini reali e osservare le attivazioni, si sintetizza l'input che massimizza l'attivazione del neurone di interesse, invertendo il flusso computazionale della rete con gradient ascent. La tecnica più diretta per capire cosa "vuole vedere" un neurone n al layer l è trovare l'input che ne massimizza l'attivazione:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} h_n^l(\mathbf{x}) - \lambda R(\mathbf{x})$$

dove $R(\mathbf{x})$ è un termine di regolarizzazione che forza la naturalezza dell'immagine. L'ottimizzazione avviene nello spazio dell'input con gradient ascent: $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla_{\mathbf{x}} h_n^l(\mathbf{x}_t)$.

Senza regolarizzazione, le immagini prodotte sono rumore avversariale — ad alta frequenza, incomprensibili. Le regolarizzazioni efficaci:

- **L2:** $R(\mathbf{x}) = \|\mathbf{x}\|^2$ — penalizza pixel con alta intensità
- **Total Variation:** $R(\mathbf{x}) = \sum_{i,j} |\nabla \mathbf{x}|_{ij}$ — favorisce immagini smooth
- **Gaussian blur periodico:** blurra ogni n step — rimuove alta frequenza
- **Jitter + crop:** ottimizza su traslazioni e crop casuali — robustezza
- **Image prior (DCGAN):** ottimizza nel latent space di un generatore già addestrato

Feature Inversion

Mahendran e Vedaldi (2015) propongono invece di ricostruire l'input da una rappresentazione intermedia $\boldsymbol{\phi}_0 = \phi(\mathbf{x}_0)$:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\phi(\mathbf{x}) - \boldsymbol{\phi}_0\|^2 + \lambda R(\mathbf{x})$$

La visualizzazione della rappresentazione $\boldsymbol{\phi}_0$ mostra le informazioni che il layer *preserva* dell'input originale — le informazioni non ricostruibili sono quelle perse dalla rappresentazione.

Superposition Hypothesis

208.1 Il Problema della Superposition

Quando una rete addestrata ha molte meno unità del numero di concetti rilevanti nel dominio, sorge una questione: come vengono codificati tutti questi concetti? La risposta teorica e sperimentale della **Superposition Hypothesis** è sorprendente. Se un layer ha n neuroni ma il mondo ha $m \gg n$ feature rilevanti, la rete non può rappresentare ogni feature in un neurone dedicato. La **Superposition Hypothesis** (Elhage et al., 2022) propone che i modelli rappresentino feature in **direzioni quasi-ortogonali** — più feature per neurone, con interferenza controllata.

Formalmente, si modella la rappresentazione come un insieme di m feature con attivazioni $f_i \in \mathbb{R}$ codificate in un layer di dimensione $n < m$. La codifica è $\mathbf{h} = \mathbf{W}\mathbf{f}$ dove $\mathbf{W} \in \mathbb{R}^{n \times m}$ con colonne $\mathbf{w}_i \in \mathbb{R}^n$. Il bias e la decodifica: $\hat{\mathbf{f}} = \mathbf{W}^T\mathbf{h} + \mathbf{b} = \mathbf{W}^T\mathbf{W}\mathbf{f} + \mathbf{b}$.

La loss di ricostruzione per feature indipendenti con importanza α_i e sparsità S_i:

$$\mathcal{L} = \sum_i \alpha_i (1 - S_i) \|\hat{f}_i - f_i\|^2$$

Il termine $(1 - S_i)$ riflette che feature sparse contribuiscono meno alla loss quando valgono zero.

208.2 Geometria degli Antipodi: Quando la Superposition è Ottimale

Il problema di ottimizzazione ha soluzioni sorprendenti. Per $n = 2$ e $m = 5$ feature identiche, la soluzione ottimale è un **pentagono regolare** — le 5 direzioni $\mathbf{w}_i$ sono equidistanti sul cerchio unitario. In generale:

> Geometria della Superposition (Elhage et al., 2022)

Per feature con importanza identica e alta sparsità, la soluzione ottimale è quella che massimizza il numero di feature rappresentate soggetta al vincolo che l'interferenza (termine off-diagonale $(\mathbf{W}^T\mathbf{W})^2_{ij}$ per $i \neq j$) rimanga sotto una soglia. Le soluzioni sono polytopes regolari: pentagono per $m = 5, n = 2$; icosaedro per $m = 6, n = 3$; ecc.

La transizione di fase: all'aumentare della sparsità S, il modello passa da zero feature rappresentate (nessuna superposition, solo feature ortogonali in n dimensioni) a superposition completa. La transizione avviene a $S^* = 1 - n/m$: sotto questa sparsità, non conviene fare superposition; sopra, conviene.

Sparse Autoencoders per Monosemanticity

209.1 Il Problema della Polysemanticity

A causa della superposition (§208), i neuroni delle reti neurali sono tipicamente **polisemantici**: un singolo neurone si attiva per concetti semanticamente eterogenei e non correlati — ad esempio, lo stesso neurone potrebbe rispondere a 'Napoleone', 'pizza' e 'codice Python' in un LLM. Questo rende l'interpretazione diretta delle attivazioni quasi impossibile. Gli **Sparse Autoencoders (SAE)** propongono una soluzione: decomporre le attivazioni polisemantiche in una base di feature monosemantiche e sparse, apprendendo un dizionario sovrapposto in cui ogni entry corrisponde a un concetto interpretabile. Questo rende l'interpretazione difficile: il neurone 42 si attiva per "finestre", "banconote" e "pirati".

Gli **Sparse Autoencoders (SAE)** (Bricken et al., 2023) imparano un dizionario di feature monosemantiche decomponendo le attivazioni di un layer in una combinazione *sparsa* di direzioni interpretabili.

209.2 Architettura e Training dell'SAE

Dato il vettore di attivazioni $\mathbf{h} \in \mathbb{R}^d$ di un layer, l'SAE impara un encoder e decoder:

$$\mathbf{f} = \text{ReLU}(\mathbf{W}_{\text{enc}}(\mathbf{h} - \mathbf{b}_{\text{pre}}) + \mathbf{b}_{\text{enc}}) \in \mathbb{R}^M \quad (M \gg d)$$

$$\hat{\mathbf{h}} = \mathbf{W}_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{pre}}$$

La loss combina ricostruzione e sparsità:

$$\mathcal{L}_{\text{SAE}} = \underbrace{\|\mathbf{h} - \hat{\mathbf{h}}\|^2}_{\text{ricostruzione}} + \lambda \underbrace{\|\mathbf{f}\|_1}_{\text{sparsità L1}}$$

Il dizionario $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{d \times M}$ ha colonne normalizzate (ogni "feature direction" è un vettore unitario). L'L1 penalty forza la maggioranza degli f_i a essere zero — solo poche feature si attivano su ogni input. Un SAE con $M = 16000$ feature su un layer di dimensione $d = 512$ di GPT-2 Small può trovare feature monosemantiche interpretabili (nome proprio, punteggiatura, codice Python, ecc.).

Feature Geometry e Monosemanticity Score

Una feature è **monolineare** se i suoi top-activating esempi hanno un tema semantico coerente. Si quantifica con il **monosemanticity score**: la frequenza con cui la feature si attiva sui top-k esempi che condividono un tema comune. Feature con score > 0.8 sono considerate monosemantiche.

Circuits: Induction Heads e Copy Suppression

210.1 Il Framework dei Circuits

Un **circuit** è un sottografo del computation graph della rete che implementa un comportamento specifico — un "algoritmo" implementato nei pesi. Olah et al. (2020) e Elhage et al. (2021) identificano circuits concreti in transformer piccoli.

La tecnica di base: **activation patching** sistematico — si eseguono due forward pass (sul prompt corretto e su un prompt "clean") e si patcha ogni componente intermedia sostituendo l'attivazione del prompt clean con quella del prompt corretto. Se la performance cambia, la componente è causalmente rilevante per il comportamento.

210.2 Induction Heads: Ripetizione di Pattern

Un **induction head** è un circuit a due testa di attenzione che implementa *in-context few-shot learning*: se il contesto contiene il pattern $[A][B]\dots[A]$, l'induction head predice $[B]$. Il meccanismo:

- **Testa 1 (Previous-Token Head):** al layer 0, la testa copia il token precedente nella propria key — $k_t^{(1)} = f(\text{token}_{t-1})$. Il valore è il token corrente.
- **Testa 2 (Induction Head proper):** al layer 1, le query di ogni token t atendono alle key prodotte dalla Testa 1. La query $q_t = f(\text{token}_t)$ cerca tra le key $k_s^{(1)}$ — trova s tale che $\text{token}_{s-1} = \text{token}_t$ (il token precedente uguale al corrente). Quindi copia il valore in posizione s — cioè il token che seguì l'ultima occorrenza di token_t.

$$\text{Induction Head Score} = \mathbb{E}[\text{attenzione dalla posizione } t \text{ alla posizione } t - L + 1]$$

dove L è la lunghezza del pattern ripetuto. Testa con score > 0.4 sono considerate induction heads. Sorprendentemente, *tutti* i transformer imparano induction heads — sono un meccan-

ismo universale emergente.

210.3 Copy Suppression e Negative Name Mover Heads

Un fenomeno meno intuitivo: alcune testa di attenzione hanno valori negativi nella direzione del nome — **sopprimono** la copia diretta di un token. Questo sembra controproducente ma serve a evitare over-confidence: quando la rete è già molto sicura di un token (alta probabilità nel residual stream), le negative name mover heads riducono leggermente la probabilità — implementando una forma di calibrazione.

Il meccanismo si quantifica con il **direct effect**: si calcola quanto ogni testa di attenzione h contribuisce direttamente al logit del token target, applicando il suo output attraverso la unembedding matrix: $\text{DE}(h) = (W_U \cdot W_O^h W_V^h)_{\text{token target}}$.

Causal Tracing e Activation Patching

211.1 Localizzare il Fatto nella Rete

Una delle domande più fondamentali sulla memoria nei LLM è: dove fisicamente nella rete viene memorizzato un fatto come “La Torre Eiffel è a Parigi”? Non in quale strato concettuale, ma in quali parametri specifici — quali pesi possono essere modificati per cambiare questo fatto senza disturbare il resto. Il **causal tracing** (Meng, Bau, Andonian e Belinkov, 2022, ROME) fornisce una metodologia rigorosa per rispondere a questa domanda localizzando dove in una rete è memorizzato un fatto specifico, es. "La Torre Eiffel è a Parigi". L’esperimento:

Algoritmo: Causal Tracing — Algoritmo

1. Prompt corretto: "La Torre Eiffel è a" → target: "Parigi"

2. Prompt corrupted: sostituisci "Eiffel" con "Urandom" → p(Parigi) cade

3. Per ogni componente (l, t) (layer l, token t):

Esegui forward sul prompt corrupted

Patcha l’attivazione di (l, t) con il valore del prompt corretto

Registra il recovery di p(Parigi)

4. La mappa di recovery mostra dove è "memorizzato" il fatto

Il risultato di Meng et al.: la MLP dell'**ultimo token del soggetto** (Eiffel) nei layer mid-level (5–10 su GPT-2 XL) è la componente con il recovery più alto — lì è "memorizzato" il fatto. Le MLP si comportano come lookup tables di fatti fattuali.

211.2 ROME: Rank-One Model Editing

Conoscendo la localizzazione, si può **modificare chirurgicamente** un fatto senza ri-training. ROME modifica i pesi MLP $\mathbf{W}^{FC}$ al layer l^* con un update rank-1:

$$\hat{\mathbf{W}} = \mathbf{W} + \frac{(\mathbf{v}^* - \mathbf{W}\mathbf{k}^*)\mathbf{k}^{*T}}{\mathbf{k}^{*T}\mathbf{C}^{-1}\mathbf{k}^*}\mathbf{C}^{-1}$$

dove $\mathbf{k}^*$ è la key activation al soggetto, $\mathbf{v}^*$ è il nuovo value target, e $\mathbf{C} = \mathbb{E}[\mathbf{k}\mathbf{k}^T]$ è la Gram matrix delle key (pre-calcolata). Questo aggiornamento è la soluzione analitica al problema di least-squares vincolato: $\min_{\mathbf{W}} \|\mathbf{W}\mathbf{K} - \mathbf{V}\|^2$ s.t. $\mathbf{W}\mathbf{k}^* = \mathbf{v}^*$.

Logit Lens, Tuned Lens e Residual Stream

212.1 La Struttura Residuale dei Transformer

Il **Logit Lens** (nostalgebraist 2020) e il **Tuned Lens** (Belrose et al. 2023) sfruttano una proprietà strutturale profonda dei Transformer: il residual stream come accumulatore progressivo di informazione decodificabile. In ogni Transformer, il **residual stream** $\mathbf{x}^l$ viene letto e scritto da ogni componente (attention heads, MLP) tramite residual connections, in modo che ogni layer aggiunga un contributo incrementale a una rappresentazione che cresce progressivamente in astrazione. La formula esatta:

$$\mathbf{x}^{l+1} = \mathbf{x}^l + \text{Attn}^l(\mathbf{x}^l) + \text{MLP}^l(\mathbf{x}^l + \text{Attn}^l(\mathbf{x}^l))$$

Ogni layer *aggiunge* al residual stream invece di trasformarlo — la rappresentazione finale è la somma dei contributi di tutti i layer. Questo permette di "leggere" il residual stream a ogni layer usando la unembedding matrix $\mathbf{W}_U$.

212.2 Logit Lens e Tuned Lens

Il **Logit Lens** (nostalgebraist, 2020): si applica $\mathbf{W}_U$ al residual stream di layer intermedi per vedere come la predizione evolve attraverso i layer:

$$\text{LogitLens}(l) = \text{softmax}(\mathbf{W}_U \cdot \text{LayerNorm}(\mathbf{x}^l))$$

Tipicamente, i layer iniziali producono predizioni casuali; dai layer mid-level si vede la "semantica" emergere; nei layer finali si raffina la sintassi e il vocabolario.

Il **Tuned Lens** (Belrose et al., 2023) migliora il Logit Lens imparando una trasformazione affine $\mathbf{T}_l$ per ogni layer che mappa il residual stream intermedio allo spazio dell'ultimo layer:

$$\text{TunedLens}(l) = \text{softmax}(\mathbf{W}_U \cdot (\mathbf{T}_l \mathbf{x}^l + \mathbf{b}_l))$$

$$\mathbf{T}_l^* = \arg\min_{\mathbf{T}_l, \mathbf{b}_l} \mathbb{E}\big[D_{\text{KL}}(p^L \| \text{softmax}(\mathbf{W}_U(\mathbf{T}_l \mathbf{x}^l + \mathbf{b}_l)))\big]$$

La trasformazione $\mathbf{T}_l$ viene addestrata su dataset con loss cross-entropy verso le predizioni del modello completo — impara la "proiezione ottimale" del residual stream intermedio verso la distribuzione finale. Questo permette di tracciare con precisione come la predizione si evolve layer per layer.

Grafi Causali, d-Separazione e Markov Condition

213.1 Structural Causal Models

La **causalità strutturale**, sistematizzata da Pearl nel suo libro "Causality" (2000), fornisce il linguaggio matematico per ragionare sulle relazioni causa-effetto in modo rigoroso, separando ciò che si può imparare da dati osservazionali da ciò che richiede interventi o esperimenti. La formalizzazione centrale è lo **Structural Causal Model (SCM)** (Pearl, 2000): una tupla $\mathcal{M} = (U, V, F)$ dove U sono le variabili esogene (rumore), V sono le variabili endogene, e $F = \{f_i\}$ sono le funzioni strutturali: $V_i = f_i(\text{Pa}(V_i), U_i)$. I genitori $\text{Pa}(V_i)$ sono le cause dirette di V_i.

Il grafo causale associato G ha un arco diretto $V_j \to V_i$ se $V_j \in \text{Pa}(V_i)$. La **Markov Condition**: ogni variabile è condizionalmente indipendente dai suoi non-discendenti dati i genitori — $V_i \perp V_j | Pa(V_i)$ se V_j non è un discendente di V_i.

213.2 d-Separazione: Indipendenze nel Grafo

Due insiemi di variabili X e Y sono **d-separati** da Z nel DAG G se ogni percorso non-diretto tra X e Y è bloccato da Z. Un percorso è bloccato se:

- Contiene una **chain** $A \to B \to C$ con $B \in Z$ (mediazione bloccata)
- Contiene un **fork** $A \leftarrow B \to C$ con $B \in Z$ (causa comune bloccata)
- Contiene un **collider** $A \to B \leftarrow C$ con $B \notin Z$ e nessun discendente di B in Z (spiegazione alternativa non attivata)

  Teorema di Markov — Equivalenza Grafo-Distribuzione

Se p è Markov rispetto a G: $X \perp_G Y | Z$ (d-separazione) implica $X \perp_p Y | Z$ (indipendenza statistica). Il viceversa vale se p è **faithful** a G: le sole indipendenze sono quelle implicate dalla d-separazione.

Identificazione del DAG dalla distribuzione osservazionale

In generale, più DAG possono essere compatibili con la stessa distribuzione — formano una **Markov Equivalence Class (MEC)**. La MEC è identificata dal CPDAG (Complete Partially Directed Acyclic Graph) — un grafo con archi orientati dove l'orientamento è identificabile e non-orientati dove non lo è. L'algoritmo PC identifica il CPDAG testando indipendenze condizionali statisticamente.

Do-Calculus e Aggiustamento Backdoor/Frontdoor

214.1 L'Operatore do( ) e l'Intervento

La distinzione operazione tra osservare e intervenire è fondamentale: osservare $X = x$ condiziona su un evento già avvenuto; $do(X = x)$ descrive cosa succederebbe se si forzasse $X = x$ dall'esterno. L'operatore $do(X = x)$ di Pearl rappresenta formalmente un **intervento esterno** che fissa $X = x$ rimuovendo tutte le cause di X nel grafo. Nella notazione del SCM: $do(X = x)$ sostituisce la funzione strutturale f_X con la costante x.

La **distribuzione interventistica** $p(Y|do(X = x))$ differisce in generale dalla distribuzione osservazionale $p(Y|X = x)$ — la seconda è confounded dalle cause comuni di X e Y.

214.2 La Regola di Aggiustamento per il Backdoor

Un insieme Z soddisfa il **backdoor criterion** rispetto a (X, Y) se: (1) nessun nodo in Z è un discendente di X; (2) Z blocca tutti i percorsi backdoor tra X e Y (percorsi che entrano in X da dietro).

Backdoor Adjustment Formula (Pearl, 1995)

$$p(Y = y|do(X = x)) = \sum_z p(Y = y|X = x, Z = z) \cdot p(Z = z)$$

Se Z soddisfa il backdoor criterion, si può calcolare l'effetto causale dall'osservazione senza esperimenti — stratificando per Z.

Esempio: si vuole stimare l'effetto di un farmaco X sulla guarigione Y. Se il medico Z prescrive il farmaco più spesso ai pazienti gravi, Z è un confounder. Stratificando per Z (severità), si ottiene l'effetto causale netto.

214.3 La Regola Frontdoor

Quando le variabili confounding U sono latenti (non osservabili), la backdoor non è applicabile. Se esiste un mediatore M tale che (1) $X \to M$ non ha backdoor non-bloccate; (2) tutti i percorsi backdoor da M a Y sono bloccati da X:

$$p(Y|do(X)) = \sum_m \sum_{x'} \underbrace{p(M = m|X)}_{\text{effetto } X \to M} \underbrace{p(Y|M = m, X = x')p(X = x')}_{\text{effetto } M \to Y \text{ senza confounder}}$$

Il classico esempio di Pearl: X = fumare, M = depositi di catrame nei polmoni, Y = cancro. Il confounding U = predisposizione genetica è latente. Applicando la frontdoor si stima l'effetto causale del fumo sul cancro senza misurare U.

Controfattuali e Structural Causal Models

215.1 La Ladder of Causation di Pearl

Judea Pearl, nel suo lavoro che gli è valso il Premio Turing 2011, ha proposto una gerarchia rigorosa dei tipi di ragionamento causale che un sistema intelligente può eseguire — la cosiddetta **Ladder of Causation** o PCH (Pearl Causal Hierarchy). Questa gerarchia non è filosofica ma matematicamente precisa: ogni livello richiede assunzioni strutturali diverse sui dati e strumenti

diversi per rispondere alle domande. La distinzione è fondamentale per l'intelligenza artificiale perché definisce esattamente cosa può imparare un sistema di ML che osserva dati passivi (solo L1), cosa richiederebbe la capacità di intervenire (L2), e cosa richiederebbe la capacità di ragionamento ipotetico su mondi alternativi (L3). Pearl identifica tre livelli di ragionamento causale:

Livello	**Operazione**	**Domanda tipica**
Association (L1)	$P(Y\|X)$	Se un paziente ha sintomo X, qual è la probabilità
Intervention (L2)	$P(Y\|do(X))$	Se diamo il farmaco X, cosa succede a Y?
Counterfactual (L3)	$P(Y_x = y\|X = x', Y = y')$	Avrebbe guarito se avesse preso il farmaco? (pur no

215.2 Algoritmo Controfattuale: Abduction-Action-Prediction

Data l'osservazione $(X = x', Y = y')$, si vuole calcolare $P(Y_{do(X=x)}|X = x', Y = y')$ — cosa sarebbe successo se X fosse stato x invece di x'.

Algoritmo: Inferenza Controfattuale nel SCM

Passo 1 — Abduction: Aggiorna il prior $P(U)$ alle evidenze osservate.

$P^*(U) = P(U|X = x', Y = y')$ (posterior sulle variabili latenti)

Passo 2 — Action: Modifica il modello intervenendo su X.

Sostituisci f_X con la costante $x \rightarrow$ modello $\mathcal{M}_{do(X=x)}$

Passo 3 — Prediction: Calcola la distribuzione di Y nel modello modificato con la posterior.

$P^*_{do(X=x)}(Y) = \sum_u f_Y(\text{Pa}(Y), u) \cdot P^*(U = u)$

Il passo chiave è l'*abduction*: si inferisce il valore delle variabili esogene U (rumore, genetica, predisposizioni) che sono compatibili con ciò che si è osservato. Poi si usa quella stima di U per predire cosa sarebbe accaduto in uno scenario alternativo.

Causal Representation Learning e IRM

216.1 Out-of-Distribution Generalization come Problema Causale

La minimizzazione del rischio empirico — il paradigma fondante del machine learning moderno — ha un difetto strutturale che emerge in modo brutale quando i dati di test differiscono da quelli di training: il modello che minimizza $\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}}[\mathcal{L}(f(x), y)]$ sfrutta *qualsiasi* correlazione presente nei dati di training, incluse le correlazioni spurie che non reggono al di fuori di quella distribuzione. Questa non è una questione di overfitting in senso classico — il modello generalizza perfettamente sulla distribuzione di training — bensì una questione di *quale* struttura il modello ha appreso. Un sistema addestrato a riconoscere animali in fotografie naturali può imparare a classificare per sfondo piuttosto che per soggetto, ottenendo alta accuracy su test set i.i.d. e fallendo completamente su test set in cui le correlazioni sfondo-animale sono invertite. L'esempio classico: un modello addestrato su immagini di cammelli su sfondo deserto e mucche su sfondo verde impara a classificare per sfondo, non per animale.

La **soluzione causale** proposta da Peters, Mooij, Schölkopf e colleghi è concettualmente netta: un classificatore basato sulle *cause genuine* di Y — nel caso degli animali, l'identità dell'animale stessa — sarà per costruzione invariante alle perturbazioni del contesto spurio (lo sfondo). Le variabili causalmente rilevanti per Y mantengono una relazione stabile attraverso tutte le distribuzioni generate intervenendo sulle variabili non causali, perché la struttura causale del mondo non cambia quando cambia il contesto di acquisizione. Questo passaggio da "correlazione" a "causalità" non è filosofico ma operativo: porta a un criterio di training diverso — l'**Invariant Risk Minimization** — che forza il modello a trovare rappresentazioni la cui relazione con Y sia stabile attraverso ambienti multipli di training.

Invariant Risk Minimization: Derivazione Completa

217.1 Il Framework IRM

IRM (Arjovsky, Bottou, Gulrajani e Lopez-Paz, 2019) traduce il principio causale della sezione precedente in un obiettivo di ottimizzazione concreto: se $\Phi(x)$ cattura le cause genuine di Y, il classificatore ottimale su $\Phi(x)$ deve essere lo stesso in ogni ambiente di training, indipendentemente dalle correlazioni spurie che variano tra ambienti. IRM formalizza l'idea di trovare una rappresentazione $\Phi : \mathcal{X} \to \mathcal{H}$ tale che il classificatore ottimale w^* sia lo *stesso* attraverso tutti gli ambienti $e \in \mathcal{E}$:

$$\min_{w,\Phi} \sum_{e\in\mathcal{E}} \mathcal{R}^e(w \circ \Phi) \quad \text{s.t.} \quad w \in \arg\min_{\tilde{w}} \mathcal{R}^e(\tilde{w} \circ \Phi) \;\; \forall e \in \mathcal{E}$$

Il vincolo dice: w deve essere ottimale per ogni ambiente *simultaneamente* — nessun ambiente richiede un classificatore diverso. Se w è ottimale contemporaneamente in tutti gli ambienti, allora Φ deve catturare le feature invarianti causali (non le correlazioni spurie che cambiano tra ambienti).

Rilassamento pratico: IRM-v1

Il vincolo originale è bilevel — difficile da ottimizzare. IRM-v1 lo rilassa usando il gradiente del classificatore come penalità:

$$\min_{w=1.0,\Phi} \sum_e \mathcal{R}^e(1.0 \cdot \Phi) + \lambda \sum_e \|\nabla_{w|w=1.0} \mathcal{R}^e(w \cdot \Phi)\|^2$$

Il termine $\|\nabla_{w|w=1.0}\mathcal{R}^e\|^2$ è zero se e solo se $w = 1.0$ è ottimale per l'ambiente e — è il quadrato della condizione del primo ordine. Se la norma del gradiente è zero per tutti gli ambienti, il classificatore fisso $w = 1.0$ è simultaneamente ottimale ovunque.

Il fix $w = 1.0$ (scalare) semplifica il problema: Φ deve produrre feature tali che il classificatore lineare banale sia ottimale. Per ottenere questo, le feature devono essere già nella forma "corretta" per la classificazione invariante causale.

Neurosimbolico: Logica Differenziabile e Neural Theorem Proving

218.1 Differentiable Logic: LTN e Neural-LP

L'integrazione di reasoning simbolico con apprendimento statistico è uno dei programmi di ricerca più ambiziosi dell'AI: i sistemi simbolici sono interpretabili, componibili e capaci di generalizzazione sistematica, ma non apprendono dai dati; le reti neurali apprendono pattern complessi ma sono opache e faticano sul ragionamento strutturato. I sistemi neurosimbolici cercano di unire i vantaggi di entrambi i paradigmi, integrando ragionamento logico con apprendimento da dati attraverso architetture differenziabili che permettono di ottimizzare end-to-end un sistema che ragiona per regole. La **Logic Tensor Network (LTN)** interpreta formule di logica del primo ordine come espressioni differenziabili:

- Costanti e variabili $\rightarrow$ tensori reali $\mathbf{e} \in \mathbb{R}^d$
- Predicati $P(x) \rightarrow$ reti neurali $\sigma(\mathbf{We} + \mathbf{b}) \in [0, 1]$
- Connettivi logici $\rightarrow$ operazioni differenziabili

$$\neg P(x) \to 1 - P(x), \qquad P(x) \wedge Q(x) \to \min(P(x), Q(x)) \;\text{ o }\; P(x) \cdot Q(x)$$

$$P(x) \vee Q(x) \to \max(P(x), Q(x)) \;\text{ o }\; P(x) + Q(x) - P(x)Q(x)$$

$$\forall x\, P(x) \to \frac{1}{N} \sum_{i=1}^{N} P(x_i) \;\text{ (aggregazione su campioni)}$$

$$\exists x\, P(x) \to \max_i P(x_i)$$

La soddisfacibilità di una KB (Knowledge Base) di formule $\{\phi_1, \ldots, \phi_K\}$ viene massimizzata minimizzando $\mathcal{L}_{\text{LTN}} = -\sum_k \text{Sat}(\phi_k)$ dove $\text{Sat}(\phi_k) \in [0, 1]$ è la verità fuzzy della formula.

Neural Theorem Proving: Differentiable Forward Chaining

Neural LP (Yang et al., 2017) impara regole logiche da grafi di conoscenza. Ogni regola ha la forma $h(X, Z) \leftarrow r_1(X, Y_1) \wedge r_2(Y_1, Y_2) \wedge \cdots \wedge r_T(Y_{T-1}, Z)$. La regola viene eseguita con moltiplicazione di matrici di adiacenza differenziabile:

$$\hat{\mathbf{A}}_h = \mathbf{A}_{r_1} \mathbf{A}_{r_2} \cdots \mathbf{A}_{r_T}$$

dove $\mathbf{A}_r[s, o] = 1$ se esiste la tripla (s, r, o) nel grafo. La composizione di relazioni implementa la chain rule logica. I pesi di ogni regola vengono appresi con backpropagation attraverso le moltiplicazioni di matrici.

Scaling Laws: Kaplan vs Chinchilla

219.1 Le Leggi di Scaling di Kaplan

Kaplan et al. (2020) scoprirono che la loss di un language model segue leggi di potenza predittive rispetto al numero di parametri N, ai token di training D, e al compute C:

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

con esponenti empirici $\alpha_N \approx 0.076$, $\alpha_D \approx 0.095$, $\alpha_C \approx 0.050$ (valori originali su GPT-2 family). Le costanti N_c, D_c, C_c dipendono dall'architettura e dal dataset.

Il budget di compute $C \approx 6ND$ (FLOPs totali). Dati un budget C fisso, la domanda è: come allocarlo tra N e D per minimizzare L?

La risposta di Kaplan: scalare principalmente N mantenendo D fisso — i grandi modelli imparano più efficientemente per token. Questa conclusione portò all'era di GPT-3 (175B parametri, 300B token).

219.2 La Legge di Chinchilla: Rivisitazione

Hoffmann et al. (2022) rivisitarono la domanda con esperimenti più rigorosi (più di 400 modelli, range più ampio di N e D). Il modello di loss:

$$L(N, D) = E + \frac{A}{N^\alpha} + \frac{B}{D^\beta}$$

dove $E \approx 1.69$ è la loss irriducibile (entropia del dataset), $A = 406.4$, $B = 410.7$, $\alpha = 0.34$, $\beta = 0.28$ (fit su dati empirici). Il compute budget $C = 6ND$, quindi:

$$N^*, D^* = \arg\min_{N,D:6ND=C} L(N, D)$$

Derivando rispetto al vincolo (Lagrangiano):

$\frac{\partial L}{\partial N} = -\frac{A\alpha}{N^{\alpha+1}} = \lambda \frac{\partial (6ND)}{\partial N} = 6\lambda D$ Condizione KKT: gradiente proporzionale al vincolo

$\frac{\partial L}{\partial D} = -\frac{B\beta}{D^{\beta+1}} = 6\lambda N$ Stessa condizione per D

Dividendo: $\frac{A\alpha N}{N^{\alpha+1}} = \frac{B\beta D}{D^{\beta+1}}$, cioè $\frac{A\alpha}{N^\alpha} = \frac{B\beta}{D^\beta}$ Le due leggi di potenza si bilanciano

$N^* = G \cdot C^{a/(a+b)}$, $D^* = G' \cdot C^{b/(a+b)}$ con $a = \alpha/(\alpha+1)$, $b = \beta/(\beta+1)$ Soluzione esplicita — il rapporto ottimale D/N è proporzionale a $C^{b/(a+b)-a/(a+b)}$

$$N^*_{\text{Chinchilla}} \approx 0.5\sqrt{\frac{C}{6}}, \qquad D^*_{\text{Chinchilla}} = C/(6N^*) \approx \frac{20N^*}{1}$$

La conclusione di Chinchilla: i modelli ottimali hanno $D \approx 20N$ token per parametro — GPT-3 con 300B token era enormemente under-trained rispetto all'ottimo. Chinchilla (70B parametri, 1.4T token) supera Gopher (280B, 300B token) su quasi tutti i benchmark.

Emergent Abilities e Phase Transitions

220.1 Il Fenomeno delle Abilità Emergenti

Tra il 2020 e il 2022, man mano che GPT-3, PaLM e Chinchilla spingevano la scala a centinaia di miliardi di parametri, i ricercatori osservarono un fenomeno sconcertante: alcune capacità sembravano comparire *dal nulla*, senza progressione graduale misurabile sui modelli più piccoli. Wei et al. (2022), nel lavoro "Emergent Abilities of Large Language Models", sistematizzano questa osservazione documentando abilità che appaiono **discontinuamente** al crescere del modello — assenti nei modelli piccoli e presenti in quelli grandi, senza miglioramento graduale intermedio. I casi canonici includono l'aritmetica multi-cifra, che emerge intorno a 100B parametri su BIG-Bench, e numerosi task di ragionamento composito che restano a livello di chance fino a una soglia critica di compute per poi salire bruscamente. Esempi: aritmetica a 3 cifre emerge intorno a 100B parametri; BIG-Bench Hard tasks emergono tra 50B e 200B.

La spiegazione matematica fornita da Schaeffer, Miranda e Koyejo (2023) in "Are Emergent Abilities of Large Language Models a Mirage?" è illuminante e controintuitiva: le "emergenze" sono in larga parte un **artefatto delle metriche a soglia** utilizzate per misurarle. Se la loss sottostante cresce linearmente con il logaritmo del compute — come previsto dalle scaling laws — ma la metrica di valutazione è discontinua per costruzione (ad esempio, il task è considerato "risolto" solo se la risposta è esattamente corretta, con threshold a 0.5 di accuracy), allora si osserva una discontinuità apparente anche in presenza di un miglioramento del tutto graduale nella performance sottostante. In altri termini, la metrica introduce una non-linearità artificiale che trasforma una crescita continua in un salto discontinuo. Con metriche continue calibrate sulla difficoltà del task, le stesse abilità mostrano crescita regolare e prevedibile. Se la loss sottostante cresce linearmente ma la metrica è discontinua (es. "risolve il problema" con threshold a 0.5), si osserva discontinuità anche da un andamento liscio. Con metriche continue, le stesse abilità mostrano crescita graduale.

Phase Transitions in Stochastic Processes

Le transizioni di fase nei modelli linguistici sono analoghe alle transizioni di fase in fisica — piccoli cambiamenti nel parametro di controllo (compute) causano cambiamenti qualitativi nel comportamento. Il **percolation theory** model: un grafo causale di ragionamento si "connette" (diventa risolvibile) solo quando abbastanza nodi (concetti) sono disponibili nella rete — transizione di percolazione.

Inference-Time Compute e Test-Time Scaling

221.1 Oltre il Training Compute: Scaling all'Inferenza

Il concetto di scaling laws (Kaplan et al. 2020, Hoffmann et al. 2022) ha guidato lo sviluppo degli LLM con una visione implicita: tutta l'"intelligenza" è fissata al momento del training. OpenAI con o1 (2024) ha dimostrato che questa visione è incompleta: le leggi di scaling standard descrivono la performance in funzione del compute di training, ma la performance può essere migliorata sistematicamente anche aumentando il compute di **inferenza** — senza modificare un singolo peso del modello, lasciando che il modello "pensi più a lungo" attraverso chain-of-thought estesi e auto-verifica.

Best-of-N: si generano N risposte e si seleziona quella con il reward model score più alto:

$$\text{BoN-}N : y^* = \arg\max_{y^{(1)},\ldots,y^{(N)} \sim \pi_\theta(\cdot|x)} r(x, y^{(i)})$$

Il miglioramento di Best-of-N segue anch'esso una legge di scaling: Win rate$(N) \propto 1 - e^{-\lambda \ln N}$ — cresce logaritmicamente con N. Per raddoppiare la performance, bisogna quadrare il numero di risposte.

Chain-of-Thought e Search: invece di generare un'unica risposta, il modello "pensa" attraverso un processo di reasoning. Con T step di CoT, la complessità del problema risolvibile scala approssimativamente come il modello con T step aggiuntivi di training.

Compute-Optimal Scaling all'Inferenza (Snell et al., 2024)

Dato un budget di compute di inferenza C_{inf}, come allocarlo tra numero di token generati T e numero di tentativi N? Empiricamente, la curva Pareto-ottimale segue:

$$L_{\text{eff}}(C_{\text{inf}}) \approx L_{\text{base}} - k \log C_{\text{inf}}$$

Il miglioramento è logaritmico nel compute di inferenza — meno efficiente del training (power law) ma disponibile senza ri-training. Per task di reasoning, modelli più piccoli con più compute di inferenza spesso superano modelli più grandi con meno compute di inferenza — la "frontiera" ottimale bilanciia la dimensione del modello con il reasoning budget.

Data Scaling: Qualità, Deduplication e Curricula

222.1 Il Ruolo della Qualità dei Dati

Le leggi di scaling di Kaplan et al. (2020) e Hoffmann et al. (2022) trattano il dataset come una grandezza scalare — "X token" — astraendo dalla sua composizione interna. Ma con dati web-scale questa astrazione fallisce: Common Crawl contiene spam, duplicati, testo machine-generated, e contenuti di scarsa qualità accanto a prosa di alta qualità. Leggi di scaling classiche assumono dati i.i.d. da una distribuzione fissa; in pratica, la **qualità** del dato è essenziale. Un modello addestrato su 10B token di alta qualità può superare uno addestrato su 100B token di bassa qualità.

Le metriche di qualità includono: (1) perplexity sotto un modello già addestrato di alta qualità (bassa perplexity = testo "simile" a testo di alta qualità); (2) classificatore di qualità addestrato su Wikipedia vs Web; (3) deduplicazione (rimozione di quasi-duplicati).

Deduplicazione: MinHash e Exact Substring Matching

La deduplicazione migliora la qualità del pretraining e riduce la memorizzazione. **MinHash LSH** (Locality Sensitive Hashing): ogni documento viene rappresentato da una firma di k hash min, e due documenti con Jaccard similarity > 0.8 sono considerati duplicati:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \approx \frac{|\{i : \min_{a \in A} h_i(a) = \min_{b \in B} h_i(b)\}|}{k}$$

Lee et al. (2022) mostrano che anche la deduplicazione a livello di **exact substring** (N-gram esatti presenti più di una volta) migliora la performance — pattern di testo identici non contribuiscono informazione aggiuntiva al modello.

222.2 Curriculum Learning e Data Mixing

L'ordine con cui si vedono i dati influenza la convergenza. Il **curriculum learning** (Bengio et al., 2009) ordina i dati dalla difficoltà crescente — analogamente all'apprendimento umano. La difficoltà di un esempio si misura con la loss iniziale del modello (alta loss = esempio difficile).

Il **data mixing** per LLM: il pretraining usa una miscela ponderata di sorgenti (web, libri, codice, matematica). Il peso di ogni sorgente influenza le capability del modello. MixIng ottimale (Albalak et al., 2023): si ottimizza il mixing weight massimizzando le performance su un set di validation tasks, trattando i pesi come iperparametri.

Long-Context: RoPE Scaling e Memory Mechanisms

223.1 Rotary Position Embedding (RoPE)

L'estensione del contesto degli LLM da 2k (GPT-2) a 128k o oltre ha richiesto un ripensamento fondamentale del position encoding. Il positional embedding assoluto apprendibile non può generalizzare a posizioni mai viste durante il training; il sinusoidale di Vaswani non scala in modo controllato. RoPE (Rotary Position Embedding, Su, Lu, Pan, Murtadha, Wen e Liu 2021) propone una soluzione elegante che codifica la posizione come una rotazione nel feature space, con la proprietà chiave che il prodotto interno $\mathbf{q}_m^T \mathbf{k}_n$ dipende solo dalla *distanza relativa* $m - n$, non dalle posizioni assolute. Per le query e le key della posizione m e n:

$$\mathbf{q}_m^T \mathbf{k}_n = \mathbf{q}^T \mathbf{R}_\Theta^{n-m} \mathbf{k}$$

dove $\mathbf{R}_\Theta^m$ è una matrice di rotazione block-diagonale con rotazioni di angolo $m\theta_i$ in ogni blocco 2D. Esplicitamente per ogni coppia di dimensioni $(2i, 2i+1)$:

$$\begin{pmatrix} q_{2i} \\ q_{2i+1} \end{pmatrix} \to \begin{pmatrix} \cos(m\theta_i) & -\sin(m\theta_i) \\ \sin(m\theta_i) & \cos(m\theta_i) \end{pmatrix} \begin{pmatrix} q_{2i} \\ q_{2i+1} \end{pmatrix}$$

con $\theta_i = b^{-2i/d}$, $b = 10000$ (base). Il prodotto interno dipende solo da $m-n$ (posizione relativa) — RoPE è naturalmente relativo. La base b determina le frequenze: basse frequenze per la struttura a lungo raggio, alte frequenze per i dettagli locali.

Estensione del Contesto: YaRN e NTK-aware Scaling

Un modello addestrato con finestra L_{train} non generalizza direttamente a sequenze $L > L_{\text{train}}$ — le rotazioni per posizioni non viste sono "fuori distribuzione". **NTK-aware scaling** (bloc97, 2023): si scala la base b proporzionalmente all'estensione del contesto:

$$b' = b \cdot (L/L_{\text{train}})^{d/(d-2)}$$

Questo preserva le frequenze relative delle rotazioni, mantenendo la coerenza tra posizioni viste e non viste. **YaRN** (Peng et al., 2023 usa interpolazione adattiva delle frequenze — basse frequenze vengono scalate meno aggressivamente di quelle alte, rispettando il "budget di frequenza" della rete.

Efficient Inference: PagedAttention e vLLM

224.1 Il Problema della KV Cache

Mentre il training di grandi modelli ha sviluppato strumenti sofisticati per l'efficienza compute (ZeRO, mixed precision), l'inference su larga scala nasconde un collo di bottiglia diverso e spesso sottovalutato: la memoria. La gestione efficiente della KV cache è il collo di bottiglia critico per l'inference di LLM su larga scala — non il calcolo in sé, ma la memoria necessaria a conservare chiavi e valori di attenzione per ogni token già generato. Durante la generazione autoregressiva, ogni nuovo token richiede il ricalcolo dell'attenzione su tutti i token precedenti. La **KV Cache** memorizza le key e value già calcolate — il costo per token scende da $O(l^2)$ a $O(l)$ dove l è la lunghezza della sequenza. Ma la KV cache ha un costo di memoria significativo: per LLaMA-2 13B con 4096 token, 2 byte per elemento:

$$\text{KV Cache size} = 2 \times L_{\text{layers}} \times N_{\text{heads}} \times d_{\text{head}} \times l_{\max} \times \text{bytes} = 2 \times 40 \times 40 \times 128 \times 4096 \times 2 \approx 3.4\,\text{GB}$$

224.2 PagedAttention: Gestione a Pagine della KV Cache

Kwon et al. (2023) propongono **PagedAttention**: la KV cache non viene allocata contiguamente in memoria, ma viene gestita come le pagine della memoria virtuale nei sistemi operativi. Ogni sequenza ha una *block table* che mappa blocchi logici della KV cache a blocchi fisici in memoria (non contigui).

I vantaggi: (1) nessuna frammentazione interna — ogni blocco fisico è usato completamente; (2) **copy-on-write** per il beam search — il prefisso comune viene condiviso tra tutti i beam senza duplicazione; (3) **prefix caching** — richieste con lo stesso system prompt condividono i blocchi fisici.

$$\text{Throughput}_{\text{vLLM}} \approx \text{GPU Utilization/KV Cache waste}$$

vLLM con PagedAttention raggiunge 2–4× più throughput di HuggingFace inference con FP16 — la principale bottleneck non è il compute ma la gestione della memoria. Il **continuous batching** completa il quadro: invece di batch statici, si aggiungono nuove richieste al batch non appena una sequenza termina — GPU always busy.

Definizione Formale di Differential Privacy

225.1 La Definizione di Dwork

La **Differential Privacy (DP)** (Dwork et al., 2006) è una garanzia matematica che l'output di un meccanismo randomizzato non rivela informazioni significative su nessun singolo individuo nel dataset. Due dataset D e D' sono **vicini** ($D \sim D'$) se differiscono in esattamente un record.

Definizione — (ε, δ)-Differential Privacy

Un meccanismo randomizzato $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ è (ε, δ)-DP se per ogni coppia di dataset vicini $D \sim D'$ e per ogni sottoinsieme misurabile $S \subseteq \mathcal{R}$:

$$\mathbb{P}[\mathcal{M}(D) \in S] \leq e^{\varepsilon}\, \mathbb{P}[\mathcal{M}(D') \in S] + \delta$$

Interpretazione: un avversario che osserva l'output del meccanismo non può distinguere se il proprio record è nel dataset con probabilità superiore a e^{ε} — con probabilità di fallimento δ. Per $\delta = 0$: **pure DP**. Per $\delta > 0$: **approximate DP**.

La **sensitivity** di una query $f : \mathcal{D} \to \mathbb{R}^k$ misura quanto l'output può cambiare rimuovendo un record:

$$\Delta_2 f = \max_{D \sim D'} \|f(D) - f(D')\|_2 \qquad \text{(L2 global sensitivity)}$$

$$\Delta_1 f = \max_{D \sim D'} \|f(D) - f(D')\|_1 \qquad \text{(L1 global sensitivity)}$$

225.2 Post-Processing e Group Privacy

Proprietà Fondamentali della DP

- **Post-processing:** se $\mathcal{M}$ è (ε, δ)-DP, allora $g \circ \mathcal{M}$ è (ε, δ)-DP per qualsiasi funzione g deterministica o randomizzata. Il modello addestrato con DP-SGD è DP — qualunque inferenza ci sia sopra.
- **Group privacy:** se $\mathcal{M}$ è (ε, δ)-DP, allora per dataset che differiscono in k record, $\mathcal{M}$ è $(k\varepsilon, ke^{(k-1)\varepsilon}\delta)$-DP.
- **Basic composition:** composizione di un meccanismo $(\varepsilon_1, \delta_1)$-DP con uno $(\varepsilon_2, \delta_2)$-DP è $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP.

Meccanismo Gaussiano e Analisi della Composizione

226.1 Il Meccanismo Gaussiano

Il meccanismo di Laplace, adatto alla DP pura ($\delta = 0$), diventa inefficiente ad alta dimensione perché la sensitivity ℓ_1 cresce con la dimensione. Per funzioni ad alto output dimensionale — come i gradienti di una rete neurale — il meccanismo gaussiano offre un tradeoff molto più favorevole: usa la sensitivity ℓ_2 (norma euclidea) e garantisce $(arepsilon, \delta)$-DP con $\delta > 0$, consentendo rumori molto più bassi. Il meccanismo gaussiano aggiunge rumore gaussiano proporzionale alla sensitivity ℓ_2 e inversamente proporzionale a ε:

$$\mathcal{M}_G(D) = f(D) + \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$$

Teorema — Meccanismo Gaussiano

Per $\sigma = \frac{\Delta_2 f \sqrt{2\ln(1.25/\delta)}}{\varepsilon}$, il meccanismo $\mathcal{M}_G$ è (ε, δ)-DP con $\varepsilon \in (0, 1)$.

Derivazione (sketch). La ratio delle probabilità tra D e D':

$\frac{p_{\mathcal{M}(D)}(\mathbf{y})}{p_{\mathcal{M}(D')}(\mathbf{y})} = \exp\left(\frac{\|\mathbf{y}-f(D')\|^2 - \|\mathbf{y}-f(D)\|^2}{2\sigma^2}\right)$ Ratio di gaussiane — il numeratore si semplifica

$= \exp\left(\frac{2(f(D)-f(D'))^T(\mathbf{y}-f(D))-\|f(D)-f(D')\|^2}{2\sigma^2}\right)$ Espansione del quadrato

Per $\mathbf{y}$ con alta probabilità, il termine lineare è $\leq \Delta_2 f\|\mathbf{y} - f(D)\|/\sigma^2$, che con alta probabilità (tail gaussiana) è $\leq \varepsilon$ Concentrazione gaussiana con σ calibrato

226.2 Advanced Composition

La composizione di base accumula ε linearmente — dopo k query (ε, δ)-DP, si ha $(k\varepsilon, k\delta)$-DP. Il **Advanced Composition Theorem** (Dwork et al., 2010) riduce questo accumulo:

$$k \text{ meccanismi } (\varepsilon,\delta)\text{-DP} \rightarrow \left(\varepsilon\sqrt{2k\ln(1/\delta')} + k\varepsilon(e^{\varepsilon}-1),\ k\delta+\delta'\right)\text{-DP}$$

Per ε piccolo: il termine dominante è $O(\varepsilon\sqrt{k})$ invece di $O(\varepsilon k)$ — risparmio di $\sqrt{k}$. Questo è il teorema che giustifica l'uso di DP-SGD con molti step di training: ogni step usa un meccanismo $(\varepsilon_0, \delta_0)$-DP, e dopo T step la privacy complessiva scala come $O(\varepsilon_0\sqrt{T})$.

Rényi Differential Privacy e Moments Accountant

227.1 La Rényi Divergence e RDP

La privacy differenziale classica $(,)$-DP ha un limite pratico: tracciare con precisione il budget di privacy accumulato attraverso migliaia di composizioni — come i passi di SGD in un training profondo — produce analisi pessimistiche. La **Rényi Differential Privacy (RDP)** (Mironov, 2017) offre una formulazione alternativa che semplifica drasticamente la composizione: le garanzie si sommano linearmente, rendendo preciso il budget tracking in DP-SGD. RDP usa la divergenza di Rényi di ordine α come misura di "distanza" tra distribuzioni:

$$D_\alpha(P\|Q) = \frac{1}{\alpha-1}\log \mathbb{E}_{x\sim Q}\left[\left(\frac{p(x)}{q(x)}\right)^\alpha\right]$$

Definizione — (α, ε)-RDP

Un meccanismo $\mathcal{M}$ è (α, ε)-RDP se per ogni $D \sim D'$: $D_\alpha(\mathcal{M}(D)\|\mathcal{M}(D')) \leq \varepsilon$.

La RDP ha proprietà di composizione più pulite che la DP classica. Composizione di k meccanismi (α, ε)-RDP: il risultato è $(\alpha, k\varepsilon)$-RDP — accumulazione *lineare* senza costanti. La conversione a (ε', δ)-DP:

$$(\alpha, \varepsilon)\text{-RDP} \implies \left(\varepsilon + \frac{\log(1/\delta)}{\alpha - 1}, \delta\right)\text{-DP}$$

Il **Moments Accountant** (Abadi et al., 2016) computa la RDP del meccanismo gaussiano per il subsampled Gaussian Mechanism (usato in DP-SGD con minibatch). Per sampling rate $q = B/N$ e rumore σ:

$$D_\alpha(\mathcal{M}_{\text{subsampled-G}} \| P) \leq \frac{1}{\alpha - 1} \log\left((1-q)^{\alpha-1} + \binom{\alpha}{2} q^2 (1-q)^{\alpha-2} e^{\alpha(\alpha-1)/(2\sigma^2)} + O(q^3)\right)$$

Il calcolo esatto (non approssimato) del RDP-budget con il Moments Accountant dà garanzie molto più tight dei bound analitici — tipicamente 2–5× migliori nella stima dell'ε effettivo.

DP-SGD: Noise Calibration e Privacy Accounting

228.1 Il Meccanismo DP-SGD

DP-SGD (Abadi, Chu, Goodfellow, McMahon, Mironov, Talwar e Zhang, Google Brain 2016) è il metodo fondativo per addestrare reti neurali con garanzie formali di differential privacy, combinando il meccanismo di Gauss con il gradient clipping per rendere ogni step di SGD $(arepsilon, \delta)$-differenzialmente privato rispetto ai singoli esempi del training set. L'intuizione è semplice: la sensitivity del gradiente dipende dalla norma massima che può assumere; se si clippa ogni gradiente individuale a norma C, poi aggiungendo rumore gaussiano di deviazione standard σC, si ottiene una garanzia DP per ogni step. La composizione di tutti i passi di training fornisce il budget totale tramite il Moments Accountant. Ad ogni step:

Algoritmo: DP-SGD — Un Step

Input: batch B di dimensione B, learning rate η, clipping threshold C, noise multiplier σ

1. Per ogni sample $x_i \in B$**:**

Calcola il gradiente per-sample: $\mathbf{g}_i = \nabla_\theta \mathcal{L}(x_i, \theta)$

Clippa il gradiente: $\tilde{\mathbf{g}}_i = \mathbf{g}_i / \max(1, \|\mathbf{g}_i\|_2 / C)$

2. Aggrega con rumore gaussiano:

$$\hat{\mathbf{g}} = \frac{1}{B}(\sum_i \tilde{\mathbf{g}}_i + \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \mathbf{I}))$$

3. Aggiornamento SGD: $\theta \leftarrow \theta - \eta\hat{\mathbf{g}}$

Il **clipping** garantisce che la sensitivity L2 del gradiente aggregato sia $\Delta_2 = C/B$ (ogni sample contribuisce al massimo C/B alla norma del gradiente). Il rumore gaussiano con $\sigma \cdot C$ soddisfa il meccanismo gaussiano.

Privacy Budget Tracking

Dopo T step con batch size B, sampling rate $q = B/N$, rumore σ, il budget totale si calcola con il Moments Accountant:

$$\varepsilon_{\text{total}}(T, \sigma, q, \delta) = \min_{\alpha} \left[\frac{T \cdot \varepsilon_{\text{RDP}}(\alpha, \sigma, q) + \log(1/\delta)}{\alpha - 1} \right]$$

Il tradeoff fondamentale di DP-SGD: per mantenere ε piccolo (forte privacy), σ deve essere grande — introduce molto rumore nei gradienti. Questo degrada la convergenza. La "privacy-utility tradeoff" è fondamentale — non esiste una soluzione gratuita.

Membership Inference Attacks

229.1 Attacchi di Membership Inference

Quando un'organizzazione addestra un modello su dati sensibili — cartelle cliniche, transazioni, comunicazioni private — e lo distribuisce come API, emerge una domanda di sicurezza: osservando solo le predizioni, è possibile capire se un individuo era nel training set? Un **Membership Inference Attack (MIA)** cerca esattamente questo: determinare se un record specifico x era nel training set di $\mathcal{M}$, osservando solo il comportamento del modello. Shokri et al. (2017) dimostrano per primi che i modelli ML rivelano informazioni di membership attraverso le loro predizioni: i modelli tendono a produrre distribuzioni di probabilità più confidenti e a loss più bassa sui dati di training rispetto a dati non visti — effetto dell'*overfitting*, anche parziale, che è sfruttabile come segnale discriminante.

L'attacco di Shokri: si addestrano "shadow models" $\mathcal{M}_1, \ldots, \mathcal{M}_k$ che imitano il modello target, usando dataset con composizione nota. Per ogni shadow model si creano esempi positivi (nel training) e negativi (non nel training) con le relative distribuzioni di output. Si addestra un classificatore di attacco su questi esempi:

$$\text{Attacco}(x, \mathcal{M}) = \text{Classifier}(\mathcal{M}(x)) \in \{\text{member}, \text{non-member}\}$$

LiRA: Likelihood Ratio Attack

Carlini et al. (2022) propongono un attacco più principled basato sul rapporto di verosimiglianza. Si campionano modelli $\mathcal{M}_{-x}$ (addestrati senza x) e si calcola il likelihood ratio:

$$\Lambda(x) = \frac{p(\text{loss}(x, \mathcal{M}) | \mathcal{M} \text{ addestrato con } x)}{p(\text{loss}(x, \mathcal{M}) | \mathcal{M} \text{ addestrato senza } x)}$$

x è classificato come member se $\Lambda(x) > 1$. LiRA è molto più accurato degli attacchi basati su shadow models — la soglia di thresholding è ottimale (Neyman-Pearson). LiRA con 256 shadow models raggiunge $> 70\%$ TPR all'1% FPR su CIFAR-10 — i modelli rivelano membership significativamente.

Federated Learning con Garanzie di Differential Privacy

230.1 DP-FedAvg

Il Federated Learning (§95, Vol. II) rappresenta un passo importante verso la privacy: i dati rimangono sui dispositivi dei client e non vengono mai trasmessi al server. Tuttavia, questa protezione è necessaria ma non sufficiente. Lavori pionieristici di Zhu et al. (2019) e Geiping et al. (2020) hanno dimostrato che gli *update del modello* — i gradienti o le differenze di peso inviate al server — possono rivelare informazioni sensibili sui dati locali attraverso tecniche di **gradient inversion**: dato il gradiente $abla_h eta \mathcal{L}(f_h eta, x, y)$, è in molti casi possibile ricostruire (x, y) con qualità sorprendente. Questa vulnerabilità è nota come gradient leakage. DP-FedAvg (McMahan et al., Google 2018) affronta il problema alla radice aggiungendo privacy differenziale formale al protocollo FedAvg, garantendo che l'aggregatore non possa estrarre informazioni sui singoli record anche osservando tutti gli update del modello.

Ogni client clippa il proprio update locale e aggiunge rumore prima di inviarlo all'aggregatore:

$$\tilde{\mathbf{u}}_k = \text{clip}(\mathbf{u}_k, C) + \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \mathbf{I})$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\eta}{K} \sum_{k=1}^{K} \tilde{\mathbf{u}}_k$$

La garanzia DP è rispetto ai *dati dei singoli client* (user-level DP) — ma la sensitivity è la norma dell'intero update del modello, che può essere molto grande. La sfida principale: con modelli grandi, il clipping deve essere aggressivo per mantenere la sensitivity controllata, degradando significativamente la convergenza.

Shuffling amplification: se i record dei client vengono "mischiati" da un intermediario fidato prima dell'aggregazione (shuffle model), la privacy migliora di un fattore $O(1/n)$ — il shuffling anonimizza la provenienza dei record. Il meccanismo shuffle permette di raggiungere $\varepsilon = 1$ con σ significativamente più piccolo di DP-FedAvg standard.

Rappresentazioni Audio: STFT, Mel Spectrogram e MFCC

231.1 Short-Time Fourier Transform

Un segnale audio $x[t]$ è tipicamente campionato a 16 kHz–44.1 kHz — frequenza scelta per coprire l'intero spettro udibile secondo il teorema di Nyquist. Per analizzare la struttura tempo-frequenza, si divide il segnale in frame sovrapposti di breve durata e si applica la FFT su ciascuno: questo produce una rappresentazione 2D che rivela come il contenuto spettrale evolve nel tempo. La **STFT** formalizza questo:

$$X[m, k] = \sum_{n=0}^{N-1} x[mH + n] \cdot w[n] \cdot e^{-j2\pi kn/N}$$

dove m è l'indice del frame, H è il hop size, N è la window size, $w[n]$ è la finestra (Hann, Hamming), e k è il bin di frequenza. Il risultato $X[m, k] \in \mathbb{C}$ ha magnitudine $|X[m, k]|$ e fase $\angle X[m, k]$.

Lo **spettrogramma** è $|X[m, k]|^2$ — l'energia per frequenza e tempo. Tipicamente si usa lo spettrogramma in scala logaritmica $\log(1 + |X[m, k]|^2)$ per comprimere il range dinamico.

231.2 Mel Filterbank e Mel Spectrogram

La scala **mel** approssima la percezione uditiva umana — l'orecchio è più sensibile alle variazioni di frequenza nelle frequenze basse. La conversione Hz $\to$ mel:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

La filterbank mel applica $M = 80$ filtri triangolari uniformemente spaziati nella scala mel, ognuno centrato su una frequenza mel distinta. Il mel spectrogram è $\mathbf{S} \in \mathbb{R}^{M \times T}$:

$$S[m, t] = \sum_{k} H_m[k] \cdot |X[t, k]|^2$$

dove $H_m[k]$ è la risposta del filtro m al bin k. Il mel spectrogram è la rappresentazione usata quasi universalmente nei modelli di speech moderni — bilancia risoluzione frequenziale e compressione.

MFCC: Mel-Frequency Cepstral Coefficients

Gli MFCC applicano la DCT al log-mel spectrogram per decorrelate le feature:

$$c[n, t] = \sum_{m=0}^{M-1} \log S[m, t] \cos\left(\frac{\pi n (m + 0.5)}{M}\right), \quad n = 0, \ldots, C - 1$$

I coefficienti MFCC catturano l'involucro spettrale del suono — la forma generale della distribuzione delle frequenze. Sono stati la rappresentazione standard per ASR per decenni, prima dei modelli end-to-end che operano direttamente su mel spectrogram o waveform.

Vocoder Neurali: HiFi-GAN e BigVGAN

232.1 Il Problema del Vocoder

La sintesi vocale (TTS) moderna è tipicamente un pipeline in due stadi: un modello acustico converte testo in mel spectrogram, e un **vocoder** converte il mel spectrogram in waveform audio a piena risoluzione (22.050 o 44.100 Hz). Quest'ultimo passo è non triviale: invertire la STFT perde la fase del segnale (informazione distrutta dalla magnitudine) e le variazioni

subframe. I vocoder classici come Griffin-Lim stimano la fase iterativamente con costosi cicli di ottimizzazione che producono artefatti metallici caratteristici. I vocoder neurali imparano la mappatura mel-spectrogram $\rightarrow$ waveform direttamente dai dati, senza assunzioni sulla fase, ottenendo qualità audio molto superiore.

232.2 HiFi-GAN: Generazione di Alta Fedeltà

HiFi-GAN (Kong et al., 2020) usa una GAN con un generatore basato su convoluzioni dilatate con **multi-receptive field fusion (MRF)**. Il generatore processa il mel spectrogram con transposed convolutions (upsampling) e poi applica blocchi residuali con kernel di diversa dimensione:

$$G(\mathbf{S}) = \text{Tanh}\left(\text{Conv}\left(\sum_{i} \text{ResBlock}_i(\text{Upsample}^k(\mathbf{S}))\right)\right)$$

La MRF somma i contributi di blocchi con kernel $[3, 7, 11]$ per ogni layer — cattura struttura a diverse scale temporali. La discriminazione avviene su tre scale temporali separate (discriminatori multi-scale) e su feature multi-periodo (discriminatori multi-period con $p \in \{2, 3, 5, 7, 11\}$ — patterns periodici nel suono).

La loss del discriminatore:

$$\mathcal{L}_D = \mathbb{E}[(D(x) - 1)^2 + (D(G(s)))^2]$$

$$\mathcal{L}_G = \mathbb{E}[(D(G(s)) - 1)^2] + \lambda_{\text{fm}}\mathcal{L}_{\text{fm}} + \lambda_{\text{mel}}\|\mathbf{S} - \hat{\mathbf{S}}\|_1$$

dove $\mathcal{L}_{\text{fm}}$ è la feature matching loss (L1 sulle feature intermedie del discriminatore) — cruciale per la stabilità del training. HiFi-GAN genera waveform a 22.05 kHz quasi indistinguibili dall'originale (MOS $> 4.4/5$) in tempo reale su CPU.

Automatic Speech Recognition: CTC e RNN-Transducer

233.1 Connectionist Temporal Classification (CTC)

CTC (Graves et al., 2006) risolve il problema dell'allineamento tra sequenza audio $\mathbf{x}$ (lunghezza T) e trascrizione $\mathbf{y}$ (lunghezza $U \ll T$) senza allineamento esplicito. Si introduce un token speciale $\varnothing$ (blank) e si somma su tutti i possibili allineamenti:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\mathbf{y})} p(\boldsymbol{\pi}|\mathbf{x}) = \sum_{\boldsymbol{\pi}} \prod_{t=1}^{T} p(\pi_t|\mathbf{x})$$

dove $\mathcal{B}$ è la funzione di collasso che rimuove blank e caratteri consecutivi ripetuti: $\mathcal{B}(\text{aa}\varnothing\text{bb}\varnothing) = ab$.

Il calcolo di $p(\mathbf{y}|\mathbf{x})$ si fa con un algoritmo forward-backward. La forward variable $\alpha_t(u) = p(\mathbf{y}_{1:u}|\mathbf{x}_{1:t})$ con ricorrenza:

$$\alpha_t(u) = [\alpha_{t-1}(u) + \alpha_{t-1}(u-1) \cdot \mathbf{1}[y_u \neq y_{u-2}]] \cdot p(y_u|\mathbf{x}_t)$$

$$+\alpha_{t-1}(u) \cdot p(\varnothing|\mathbf{x}_t)$$

Il gradiente rispetto ai logit di ogni frame è calcolato analiticamente tramite la backward variable $\beta_t(u)$. CTC assume **conditional independence** delle emissioni: $p(\boldsymbol{\pi}|\mathbf{x}) = \prod_t p(\pi_t|\mathbf{x})$ — ogni frame decide indipendentemente. Questa è un'assunzione semplificante ma molto efficace nella pratica.

233.2 RNN-Transducer: Modello Autoregressivo per ASR

Il **RNN-T** (Graves, 2012) rilassa l'assunzione di indipendenza di CTC — mantiene un modello del linguaggio autoregressivo (prediction network) che interagisce con l'encoder acustico. Il joint network combina:

$$h(t, u) = \tanh(\mathbf{W}_{\text{enc}}\mathbf{f}_t + \mathbf{W}_{\text{pred}}\mathbf{g}_u + \mathbf{b})$$

$$p(y|\mathbf{x}, t, u) = \text{softmax}(\mathbf{W}_h h(t, u) + \mathbf{b}_h)$$

dove $\mathbf{f}_t$ è l'encoding acustico al frame t e $\mathbf{g}_u$ è l'encoding del linguaggio dopo aver emesso u token. L'output può essere un token del vocabolario o il blank $\varnothing$ — il blank avanza il puntatore t (più audio); i token avanzano il puntatore u (più output).

La loss è analoga a CTC ma con il modello del linguaggio:

$$\log p(\mathbf{y}|\mathbf{x}) = \log \sum_{\boldsymbol{\pi}} \prod_{(t,u)\in\boldsymbol{\pi}} p(\pi_{t,u}|\mathbf{x}, \mathbf{y}_{<u})$$

L'RNN-T supera CTC sui task streaming e produce trascrizioni di qualità superiore grazie al modello del linguaggio integrato. È la base di tutti i sistemi ASR moderni per on-device (Google Pixel, Apple Watch).

Whisper: Multitask e Multilingual ASR

234.1 Architettura e Formato Multitask

La storia del riconoscimento vocale automatico (ASR) è stata a lungo dominata da sistemi specializzati per singola lingua, addestrati su dataset costosamente annotati e ottimizzati per distribuzioni acustiche specifiche. OpenAI nel 2022 ha proposto con Whisper (Radford, Kim, Xu, Brockman, McLeavey e Sutskever) un'inversione radicale di questa logica: anziché costruire sistemi specializzati, addestrare un singolo modello encoder-decoder su **680.000 ore di audio web-scraped in 99 lingue**, accettando che la qualità delle annotazioni sia eterogenea in cambio della scala. Questa scommessa sulla quantità di dati eterogenei si rivela vincente: Whisper raggiunge o supera sistemi specializzati su molti benchmark, mostrando una robustezza straordinaria a condizioni acustiche difficili (rumore, accenti, musica di sottofondo) grazie alla varietà del corpus di training. L'encoder processa il mel spectrogram con 2 layer di convoluzioni e poi un encoder Transformer. Il decoder genera token autoregressivamente condizionato sull'audio encoder.

La chiave del design: **format tokens** che specificano il task nel prompt del decoder:

$$[\text{SOT}][\text{lang_token}][\text{task_token}][\text{notimestamps?}] \rightarrow \text{trascrizione}$$

I token speciali includono: ⟨|en|⟩ (English), ⟨|it|⟩ (Italian), ⟨|transcribe|⟩ (trascrizione), ⟨|translate|⟩ (traduzione verso inglese). Questo permette a un singolo modello di fare ASR in 99 lingue e traduzione simultanea.

Robustezza e Limiti

Whisper mostra robustezza eccezionale a rumore di fondo, accenti e condizioni acustiche diverse — la varietà dei 680k ore di training copre molti contesti. I limiti: hallucinazione (genera trascrizioni plausibili anche per audio vuoto), bias verso la lingua inglese (i 680k ore non sono bilanciati), sensibilità al silenzio (può produrre loop di testo su audio con lunghi silenzi).

TTS: FastSpeech 2 e VITS

235.1 FastSpeech 2: Non-Autoregressive TTS

I sistemi TTS autoregressivi come Tacotron 2 e WaveNet producono audio di alta qualità ma sono intrinsecamente lenti: la generazione di ogni frame dipende da tutti i precedenti, impedendo la parallelizzazione. La risposta della comunità è stata lo sviluppo di architetture **non-autoregresssive** che generano l'intera sequenza in parallelo. **FastSpeech 2** (Ren et al., 2021) genera mel spectrogram da testo in modo non-autoregressivo — tutto il mel spectrogram in un singolo passo, rendendo la sintesi 270× più veloce di Tacotron 2. La pipeline:

$$\text{Testo} \xrightarrow{\text{encoder}} \mathbf{H} \xrightarrow{\text{LR-predictor}} \tilde{\mathbf{H}} \xrightarrow{\text{var-adapter}} \hat{\mathbf{H}} \xrightarrow{\text{decoder}} \text{Mel}$$

Il **Length Regulator** con **duration predictor** trasforma la sequenza di fonemi $\mathbf{H} \in \mathbb{R}^{N \times d}$ in una sequenza allineata con il mel $\tilde{\mathbf{H}} \in \mathbb{R}^{T \times d}$ replicando ogni fonema per il numero di frame predetto. I **variance adaptors** aggiungono pitch e energy predetti per frame — controllano l'espressività della voce sintetizzata.

235.2 VITS: End-to-End TTS con VAE e GAN

VITS (Kim et al., 2021) è il sistema TTS end-to-end allo stato dell'arte — genera waveform direttamente dal testo in un singolo modello, eliminando il vocoder separato. L'architettura combina VAE e GAN:

$$\text{Testo} \xrightarrow{\text{encoder+flow}} z \sim q_\phi(z|\mathbf{x}) \xrightarrow{\text{decoder}} \text{waveform}$$

Il flow-based posterior encoder: dato il mel spectrogram target $\mathbf{x}$ e il testo, il posterior $q_\phi(z|\mathbf{x}, c)$ è gaussiano con parametri prodotti da un normalizing flow f: $q_\phi = \mathcal{N}(f(\mathbf{x}), \sigma^2)$. Il prior è un'altra gaussiana $p(z|c)$ condizionata sul testo. La loss combina ELBO + GAN adversarial:

$$\mathcal{L}_{\text{VITS}} = \mathcal{L}_{\text{recon}} + D_{\text{KL}}(q\|p) + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{\text{fm}}$$

Il duration predictor in VITS usa una stochastic duration predictor — modella l'incertezza nella durata dei fonemi, producendo sintesi più naturale con variazione ritmica.

Codec Language Models: AudioLM, MusicGen e EnCodec

236.1 EnCodec: Neural Audio Codec

Applicare i transformer alla generazione audio presenta una sfida di tokenizzazione fondamentale: un secondo di audio campionato a 22.050 Hz produce 22.050 campioni — una sequenza immensamente più lunga di quella testuale equivalente, incompatibile con la complessità quadratica dell'attenzione. Per generare audio di alta qualità con LLM, è necessario prima comprimere il waveform in sequenze di token discreti molto più corte. **EnCodec** (Défossez et al., Meta AI 2022) risolve questo: è un codec audio neurale che comprime waveform a bitrate molto bassi con alta qualità percettiva. L'architettura: encoder convoluzionale → quantizzatore residuale (RVQ) → decoder convoluzionale. Il Residual Vector Quantization (RVQ) quantizza sequenzialmente il residuo:

$$\mathbf{z}_0 = \text{Encoder}(\mathbf{x})$$

$$\mathbf{z}_{q,k} = \text{VQ}_k(\mathbf{z}_0 - \sum_{j<k} \hat{\mathbf{z}}_j), \quad \hat{\mathbf{z}}_k = \text{decode-from-codebook}(\mathbf{z}_{q,k})$$

Con 8 quantizers e codebook size 1024 (10 bit ciascuno), EnCodec raggiunge 1.5 kbps mantenendo audio intelligibile — 50× compressione rispetto a 44.1 kHz PCM. Il token rate è 75

token/secondo per 8 livelli di RVQ = 600 token/secondo totali. Questo è il tasso che AudioLM e MusicGen processano come "audio tokens".

236.2 MusicGen: Generazione Musicale Condizionata

MusicGen (Copet et al., 2023) usa un decoder-only Transformer che genera token EnCodec autoregressivamente, condizionato su testo o melodia. Il punto chiave: i codebook multipli di RVQ producono $Q = 4$–8 token per timestamp — come generarli?

MusicGen usa la tecnica dei **delay patterns**: invece di generare tutti i Q token di ogni timestamp prima di passare al successivo, si sfasano i codebook di 1 step ciascuno:

$$\text{Timestep } t\text{: genera token } [c_1(t), c_2(t-1), c_3(t-2), \ldots, c_Q(t-Q+1)]$$

Il transformer genera un token per step del codebook 1, poi il codebook 2 del timestep precedente, ecc. — tutti i Q codebook si "allineano" grazie allo sfasamento, producendo audio coerente con un singolo modello autoregressivo semplice. L'alternativa (generazione parallela di tutti i Q token) richiede modelli più complessi.

Diffusion Policies: Score Matching per Behavioral Cloning

237.1 Behavioral Cloning come Stima di Densità

L'imitazione di un esperto umano in robotica (behavioral cloning) è tradizionalmente trattata come regressione: dato uno stato s, predire l'azione a dell'esperto. Il problema: la distribuzione delle azioni è tipicamente **multimodale** — lo stesso stato può richiedere azioni diverse (scegliere tra due oggetti, scegliere in quale direzione girare).

Le **Diffusion Policies** (Chi et al., 2023) modellano la distribuzione delle azioni condizionata sullo stato come un processo di diffusione, evitando il problema della multimodalità:

$$\pi(\mathbf{a}|\mathbf{o}) = p_\theta(\mathbf{a}|\mathbf{o}) \approx q_{\text{data}}(\mathbf{a}|\mathbf{o}) \qquad \text{(distribuzione target)}$$

Il modello apprende $\boldsymbol{\epsilon}_\theta(\mathbf{a}^k, k, \mathbf{o})$ — la rete di denoising che predice il rumore aggiunto all'azione al livello k del processo di diffusione, condizionata sull'osservazione $\mathbf{o}$ (immagine + stato del robot).

237.2 Architettura e Conditioning

La rete di denoising usa un'architettura U-Net 1D per azioni temporali (sequenza di azioni future $\mathbf{a}_{t:t+H}$) condizionata su osservazioni $\mathbf{o}_{t-T_o:t}$ tramite FiLM conditioning:

$$\text{FiLM}(\mathbf{h}, \mathbf{o}) = \gamma(\mathbf{o}) \odot \mathbf{h} + \beta(\mathbf{o})$$

dove $\gamma(\mathbf{o})$ e $\beta(\mathbf{o})$ sono scale e shift predetti dall'observation encoder (ViT o ResNet). L'inferenza genera l'intera sequenza di azioni future in $K = 100$ step di DDPM denoising partendo da rumore gaussiano — garantisce coerenza temporale dell'azione.

Risultati: su task di manipolazione con oggetti deformabili (piegare il bucato, impacchettare cibo), Diffusion Policy raggiunge 90% success rate contro 70% del behavior cloning standard — la capacità di catturare distribuzioni multimodali è cruciale per l'imitazione fedele.

Vision-Language-Action Models: RT-1 e RT-2

238.1 RT-1: Robotics Transformer per Controllo

La robotica ha a lungo operato con pipeline modulari: perception separato da planning, planning separato da control, con interfacce ingegneristiche tra i moduli. I Vision-Language-Action Models (VLA) propongono una rottura radicale: un unico modello transformer addestrato end-to-end che mappa osservazioni visive e istruzioni linguistiche direttamente in azioni di basso livello, superando la necessità di rappresentazioni intermedie esplicite. RT-1 (Brohan et al., Google Robotics 2022) è il primo sistema su larga scala che dimostra la fattibilità di questo approccio: un Transformer che mappa direttamente immagini + istruzioni testuali $\rightarrow$ azioni motorie discrete del robot. L'architettura elabora l'immagine con EfficientNet-B3 e testo con Sentence-BERT, poi una sequenza di 6 frame viene processata da un FiLM-conditioned Transformer decoder.

Le azioni discrete tokenizzate: ogni azione ha 7 dimensioni (posizione end-effector Δxyz, orientamento Δrpy, presa). Ogni dimensione viene discreta in 256 bin $\rightarrow$ l'azione è una sequenza di 8 token (7 dim + gripper stato). Il modello genera le azioni autoregressivamente come un language model genera testo.

238.2 RT-2: Vision-Language Models come Robotics Policies

RT-2 (Brohan et al., 2023) usa direttamente un VLM (PaLM-E o PaLI-X) pre-addestrato su dati web come policy robotica. Le azioni vengono tokenizzate come testo e il modello è fine-tuned end-to-end su dati robotici mantenendo il pre-training web.

Il trucco chiave: le azioni vengono rappresentate come stringhe numeriche — "move arm to [255] [128] [64] ..." dove ogni numero è un bin discretizzato. Il VLM già sa come generare numeri, quindi il fine-tuning per le azioni è naturale.

Il **chain-of-thought per robotica**: RT-2 può ragionare sulle azioni in linguaggio naturale prima di eseguirle. Prompt: "move the object that is not a fruit to the left" $\rightarrow$ il modello "pensa" "the objects are a banana and a cup; cup is not a fruit; move cup left" $\rightarrow$ genera l'azione di muovere la tazza. La generalizzazione emergente da web-scale pretraining è sorprendente — RT-2 gestisce oggetti mai visti in training.

Model Predictive Control Neurale

239.1 MPC: Ottimizzazione Online dell'Azione

Nelle applicazioni di controllo in cui la sicurezza è critica — robotica industriale, guida autonoma, sistemi di processo — apprendere una policy end-to-end come in DRL ha un limite fondamentale: la policy è una scatola nera difficile da verificare e da adattare a nuovi vincoli senza riaddestrare. Il **Model Predictive Control (MPC)** offre un'alternativa interpretabile: anziché apprendere una policy che mappa stati in azioni, si mantiene un modello esplicito del sistema $\hat{f}$ e si risolve un problema di ottimizzazione a ogni passo per l'azione ottimale su un orizzonte finito. Con un modello neurale $\hat{f}_\theta$ appreso dai dati, MPC combina la flessibilità del deep learning con la pianificazione esplicita e i vincoli di sicurezza. Non apprende una policy direttamente — usa un modello del mondo $\hat{f}$ per pianificare online, riottimizzando ad ogni passo. Ad ogni passo:

Algoritmo: MPC con Modello Neurale

dato stato corrente s_t, orizzonte H, modello $\hat{f}_\theta$

ottimizza: $\mathbf{a}^* = \arg\min_{a_t,\ldots,a_{t+H}} \sum_{h=0}^{H} c(s_{t+h}, a_{t+h})$

s.t. $s_{t+h+1} = \hat{f}_\theta(s_{t+h}, a_{t+h})$

esegui solo a_t^*, ricevi s_{t+1}, ripeti

L'ottimizzazione delle azioni si fa con metodi basati sul campionamento (Cross-Entropy Method, MPPI) o su gradient descent attraverso il modello differenziabile.

MPPI: Model Predictive Path Integral

MPPI (Williams et al., 2017) campiona K sequenze di perturbazioni $\boldsymbol{\epsilon}^{(k)} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ attorno alla sequenza nominale $\bar{\mathbf{a}}$ e seleziona la nuova sequenza nominale come media pesata esponenzialmente:

$$w^{(k)} = e^{-S(\tau^{(k)})/\lambda}, \quad \bar{\mathbf{a}} \leftarrow \frac{\sum_k w^{(k)}(\bar{\mathbf{a}} + \boldsymbol{\epsilon}^{(k)})}{\sum_k w^{(k)}}$$

dove $S(\tau^{(k)}) = \sum_h c(s_h^{(k)}, a_h^{(k)})$ è il costo della traiettoria k e λ è il parametro di temperatura. Le traiettorie con basso costo ricevono peso alto — MPPI è un importance sampling del costo. Il calcolo di tutte le traiettorie è parallelo su GPU — possibile in real-time con modelli neurali rapidi.

SLAM Neurale e Rappresentazioni 3D Implicite

240.1 iMAP e NICE-SLAM: Dense SLAM con NeRF

Lo Simultaneous Localization and Mapping (SLAM) è uno dei problemi fondamentali della robotica: costruire una mappa dell'ambiente mentre si stima la propria posizione in essa, in modo incrementale e in tempo reale. Le soluzioni classiche usano point clouds, voxel grids o occupancy grids come rappresentazione della mappa — strutture discrete che scalano male in memoria e non catturano superfici continue. **Neural SLAM** rappresenta una rottura paradigmatica: usa reti neurali implicite — specificamente NeRF — come rappresentazione continua e compatta della mappa 3D. Questa scelta porta vantaggi notevoli: la rete può essere interrogata a qualsiasi risoluzione, interpola in modo continuo tra le osservazioni, e la sua differenziabilità consente l'ottimizzazione congiunta di mappa e pose della camera con discesa del gradiente. iMAP (Sucar et al., ICCV 2021) e NICE-SLAM (Zhu et al., CVPR 2022) sono i sistemi fondativi che ottimizzano congiuntamente la rete della scena e le pose della camera. **iMAP** (Sucar et al., 2021) e **NICE-SLAM** (Zhu et al., 2022) ottimizzano congiuntamente la rete della scena e le pose della camera.

L'ottimizzazione ha due componenti: (1) **mapping** — fissa le pose, ottimizza i pesi della rete; (2) **tracking** — fissa la rete, ottimizza la posa della camera corrente minimizzando il render

error:

$$\mathcal{L}_{\text{track}} = \sum_{\mathbf{r}} \left[\|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|^2 + \|\hat{D}(\mathbf{r}) - D(\mathbf{r})\|^2 \right]$$

dove $\hat{C}(\mathbf{r})$ e $\hat{D}(\mathbf{r})$ sono il colore e la profondità renderizzati dal NeRF lungo il raggio $\mathbf{r}$, e $C(\mathbf{r}), D(\mathbf{r})$ sono i valori osservati dalla camera. Il gradiente rispetto alla posa della camera viene propagato attraverso il rendering NeRF differenziabile.

NeRF per Manipolazione: Derivazione del Rendering Differenziabile

241.1 Differenziabilità del Volume Rendering

Una delle proprietà più potenti di NeRF (Neural Radiance Fields, Mildenhall et al. 2020) è che l'intero pipeline di rendering è differenziabile rispetto ai parametri della rete neurale, consentendo l'addestramento per backpropagation a partire da semplici immagini 2D senza supervisione 3D. Questa proprietà, nota come **rendering differenziabile**, è la chiave che ha aperto la strada all'uso di NeRF in robotica e manipolazione: se il rendering è differenziabile, si può ottimizzare qualsiasi quantità che dipende dall'immagine renderizzata — inclusa la posa della camera o la configurazione del manipolatore. La formula di volume rendering NeRF:

$$C(\mathbf{r}) = \int_0^\infty T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt, \quad T(t) = e^{-\int_0^t \sigma(\mathbf{r}(s))\, ds}$$

è differenziabile rispetto ai parametri della rete θ che predicono $(\sigma, \mathbf{c})$ — i gradienti si propagano attraverso l'integrale di rendering tramite backpropagation standard. Questa proprietà è la chiave che ha aperto la strada all'uso di NeRF in robotica: se il rendering è differenziabile, si può ottimizzare qualsiasi quantità che dipende dall'immagine renderizzata — inclusa la posa della camera, la configurazione di un manipolatore, o i parametri di un controller. Il gradiente fluisce dall'immagine osservata attraverso il processo di rendering fino ai parametri della scena o della policy, senza bisogno di approssimazioni finite-difference costose. In forma discreta:

$$\hat{C} = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i, \quad T_i = \prod_{j<i} (1 - \alpha_j), \quad \alpha_i = 1 - e^{-\sigma_i \delta_i}$$

Il gradiente rispetto ai pesi della rete $\partial \hat{C}/\partial \mathbf{c}_i = T_i \alpha_i$ e $\partial \hat{C}/\partial \sigma_i = T_i \delta_i e^{-\sigma_i \delta_i} \mathbf{c}_i - \delta_i \sum_{j>i} T_j \alpha_j \mathbf{c}_j$.

Per la robotica di manipolazione: si usa il NeRF per stimare la **pose 6DoF** di oggetti. Date immagini della scena, si ottimizza la posa $(\mathbf{R}, \mathbf{t})$ dell'oggetto minimizzando la differenza tra il NeRF renderizzato e l'immagine osservata — senza fiducials o marker artificiali. La precisione raggiunge < 5 mm per oggetti texturati.

Protein Language Models: ESM e ProtTrans

242.1 Le Proteine come Sequenze

Il parallelismo tra linguaggio naturale e sequenze biologiche non è metaforico: è una constatazione strutturale precisa. Una proteina è una sequenza di amminoacidi da un alfabeto di $|\mathcal{V}| = 20$ token (amminoacidi standard), e il genoma umano ne codifica oltre 20.000 diverse. Come nel linguaggio naturale, la posizione e il contesto determinano il "significato" funzionale — due amminoacidi lontani nella sequenza primaria possono essere vicini nella struttura 3D e interagire funzionalmente attraverso legami non covalenti. Ma il parallelismo va più in profondità: l'evoluzione ha agito sulle sequenze proteiche come un processo di selezione linguistica, preservando pattern che codificano funzione e struttura. Questa analogia ha suggerito di applicare direttamente i **protein language models (PLM)** — transformer pre-addestrati su grandi database di sequenze proteiche — come modo per apprendere rappresentazioni biologicamente informative senza supervisione strutturale esplicita.

ESM-2 (Lin et al., 2022) è un BERT pre-addestrato su 250M sequenze proteiche con masked language modeling. L'obiettivo: dato il contesto della sequenza, predire gli amminoacidi mascherati. Con 15B parametri e training su 250M proteine di UR50, ESM-2 produce rappresentazioni che catturano struttura, funzione ed evoluzione.

242.2 Scaling dei Protein Language Models

A differenza dei LLM dove le emergent abilities appaiono intorno ai 100B parametri, nei protein LM si osservano transizioni già ai 150M parametri — il "vocabolario" evoluzionario delle proteine ha meno complessità del linguaggio umano. La perplexity sulle sequenze proteiche scende logaritmicamente con il numero di parametri, ma la qualità delle strutture predette (misurata da TM-score) mostra discontinuità intorno ai 650M parametri.

ESMFold: Struttura dalla Sequenza in 1 Secondo

ESMFold (Lin et al., 2022) aggiunge una struttura folding testa a ESM-2 — predice la struttura 3D direttamente dalla sequenza senza allineamenti multipli (MSA). La velocità: 1 secondo per proteina di lunghezza media, contro i minuti di AlphaFold2. La qualità: 60–70% di TM-score su proteine isolate, inferiore ad AlphaFold2 ma sufficiente per molte applicazioni di screening.

AlphaFold2: Evoformer Completo e Invariant Point Attention

243.1 Multiple Sequence Alignment e Rappresentazione Duale

AlphaFold2 (Jumper et al., 2021) usa due rappresentazioni duale del problema di folding:

- **MSA representation** $\mathbf{m} \in \mathbb{R}^{S \times L \times c_m}$: S sequenze omologhe, L posizioni, c_m features — cattura le covarianze evolutive
- **Pair representation** $\mathbf{z} \in \mathbb{R}^{L \times L \times c_z}$: features per ogni coppia di residui (i, j) — cattura le relazioni strutturali

L'**Evoformer** è il blocco principale che aggiorna entrambe le rappresentazioni in modo co-evolutivo.

243.2 Evoformer: Outer Product Mean e Triangle Attention

La comunicazione dall'MSA alla rappresentazione pair avviene tramite **outer product mean**:

$$\mathbf{z}_{ij} \leftarrow \mathbf{z}_{ij} + \text{Linear}\left(\frac{1}{S} \sum_{s} \mathbf{m}_{si} \otimes \mathbf{m}_{sj} \right)$$

Il prodotto esterno $\mathbf{m}_{si} \otimes \mathbf{m}_{sj} \in \mathbb{R}^{c_m \times c_m}$ cattura la covarianza tra due posizioni nella stessa sequenza. La media su S sequenze estrae la covarianza evolutiva — le posizioni che co-evolvono tendono ad essere spazialmente vicine nella struttura.

La **Triangle Attention** aggiorna la pair representation sfruttando la struttura triangolare delle distanze 3D. Per ogni tripla di residui (i, j, k), la distanza d_{ij} è vincolata da d_{ik} e d_{kj} (disuguaglianza triangolare):

$$\mathbf{z}_{ij}^{\text{out}} \leftarrow \text{Attn}(\mathbf{q}_{ij}, \{\mathbf{k}_{ik}\}_k, \{\mathbf{v}_{ik}\}_k) \cdot \text{gate}(\mathbf{z}_{ij})$$

$$\text{bias}_{ijk} = \mathbf{W}_b \mathbf{z}_{kj} \qquad \text{(attenzione modolata da terzo lato)}$$

La triangle attention "per lato" (starting node) e "per lato" (ending node) vengono applicate in alternanza — lo spazio delle coppie si aggiorna considerando tutte le possibili triplette.

243.3 Invariant Point Attention (IPA)

Il Structure Module di AlphaFold2 genera le coordinate 3D tramite aggiornamenti iterativi di frame rigidi $(R_i, \mathbf{t}_i)$ — la rotazione e traslazione che posizionano ogni residuo nello spazio 3D. L'**IPA** è un meccanismo di attenzione che opera su questi frame mantenendo l'equivarianza E(3):

$$a_{ij} \propto \sum_h \left[q_i^{(h)T} k_j^{(h)} + \sum_p w_{hc} \left(-\frac{\gamma}{2} \| T_i q_i^{(h,p)} - T_j k_j^{(h,p)} \|^2 \right) \right]$$

Il secondo termine usa i query/key nella coordinata del frame locale del residuo — proiettati nel frame globale tramite la trasformazione rigida T_i. La differenza $\|T_i q_i - T_j k_j\|^2$ è invariante a rotazioni e traslazioni globali del sistema, garantendo l'equivarianza E(3) dell'intera architettura.

Modelli Generativi per Drug Discovery

244.1 Generazione di Molecole come Grafi

Il drug discovery tradizionale si basa su screening di librerie chimiche esistenti: si testano milioni di composti già noti cercando quelli con l'attività biologica desiderata. I modelli generativi offrono un paradigma alternativo: progettare de novo molecole con proprietà target specificate, esplorando lo spazio chimico in modo guidato. Le molecole sono rappresentate naturalmente come grafi — atomi come nodi, legami come archi — e la generazione di molecole *valide* sia dal punto di vista della chimica organica (valenza, aromaticità) che delle proprietà biologiche (ADMET: assorbimento, distribuzione, metabolismo, escrezione, tossicità) è un problema di generazione di grafi vincolata. Due approcci principali:

REINFORCE per grafici (You et al., 2018): si usa policy gradient per aggiungere nodi e archi iterativamente. La reward è la proprietà desiderata della molecola (es. drug-likeness, binding affinity). La policy π_θ decide quale atomo/legame aggiungere ad ogni step:

$$\nabla_\theta J = \mathbb{E}_{G \sim \pi_\theta} [R(G) \nabla_\theta \log \pi_\theta(G)]$$

dove $R(G)$ è la reward della molecola generata G (calcolata con un docking simulator o un QSAR model).

244.2 Flow Matching per Molecular Generation

I modelli di diffusione e flow matching per molecole lavorano nello spazio delle coordinate atomiche 3D. Il modello deve essere E(3)-equivariante — le rotazioni e traslazioni della molecola non cambiano le sue proprietà fisico-chimiche.

Equivariant Flow Matching (Song et al., 2023): si definisce un flusso $p_t(\mathbf{x})$ da una gaussiana isotropa a $t = 0$ alla distribuzione molecolare $t = 1$. Il vettore velocity è equivariante:

$$\mathbf{v}_\theta(\mathbf{x}_t, t) = \sum_{\mathbf{R},\mathbf{b}} \mathbf{R}^T \tilde{\mathbf{v}}(\mathbf{R}\mathbf{x}_t + \mathbf{b}, t) \qquad \text{(equivarianza per definizione)}$$

In pratica si ottiene l'equivarianza usando EGNN (§77, Vol. II) come rete velocity. Il training usa la Conditional Flow Matching loss con percorsi ottimali (OT-FM, §78, Vol. II) — minimizza la distanza tra le molecole del training e quelle generate.

DNA Language Models e Genomica

245.1 Sequenze Genomiche come Linguaggio

La genomica computazionale ha sempre trattato le sequenze di DNA come stringhe simboliche, applicando metodi di allineamento basati su similarità evoluzionistica. L'avvento dei transformer ha aperto una prospettiva radicalmente diversa: il DNA è una sequenza su un alfabeto di quattro simboli (A, T, C, G) e il genoma umano, con i suoi $\sim 3 \times 10^9$ paia di basi, è la "frase" più lunga che esista in natura — circa due volte la lunghezza dei corpora testuali su cui sono addestrati i più grandi LLM. Trattare il DNA come un linguaggio e addestrare transformer su sequenze genomiche produce rappresentazioni che codificano struttura della cromatina, siti di binding di fattori di trascrizione, e regioni regolatorie — informazioni biologiche profonde che emergono in modo non supervisionato dalla sola statistica delle sequenze. Il pretrained su sequenze genomiche produce rappresentazioni che codificano struttura cromatina, siti di binding dei fattori di trascrizione, e varianti patogeniche.

Nucleotide Transformer (Dalla-Torre et al., 2023): addestrato su 3202 genomi umani e 850 genomi non-umani. Tokenizzazione: ogni token è un 6-mer (sequenza di 6 basi) — vocabolario di $4^6 = 4096$ token. Con questo vocabolario, l'intero genoma umano corrisponde a $\sim 5 \times 10^8$ token.

Predire la Regolazione Genica

I task downstream includono: (1) predire siti di binding dei fattori di trascrizione (TF-binding); (2) predire la cromatina accessibile (ATAC-seq); (3) predire l'espressione genica. Lo stato dell'arte: fine-tuning del Nucleotide Transformer supera architetture specifiche (DeepBind, DanQ) su 18/25 task dell'Genomic Benchmarks.

Enformer (Avsec et al., 2021: un Transformer addestrato per predire coverage di 5313 esperimenti genomici (ATAC-seq, ChIP-seq, RNA-seq) da 196kb di sequenza genomica. L'attenzione a lungo raggio è cruciale — gli enhancer (elementi regolatori) possono trovarsi a > 100kb dal gene che regolano. Enformer raggiunge $R^2 > 0.9$ per la predizione dell'espressione genica.

GNN Geometriche per Molecole: SchNet, DimeNet, PaiNN

246.1 SchNet: Interazioni Atomiche con Continuous Filter

SchNet (Schütt et al., 2017) è il primo modello GNN per proprietà molecolari che usa interazioni continue invece di fingerprint discreti. Ogni atomo i ha embedding $\mathbf{h}_i$; le interazioni dipendono solo dalla distanza interatomica $d_{ij} = \|\mathbf{r}_i - \mathbf{r}_j\|$:

$$\mathbf{v}_i = \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j \odot \underbrace{W(d_{ij})}_{\text{filter network}}$$

Il filter network $W : \mathbb{R} \to \mathbb{R}^{n_{\text{feat}}}$ è una piccola rete neurale che produce filter coefficients dipendenti dalla distanza. Expand the distance in radial basis functions (RBF):

$$e_k(d) = \exp(-\gamma(d - \mu_k)^2), \quad k = 1, \ldots, K$$

con $K = 300$ gaussiane uniformemente distribuite in $[0, d_{\max}]$ e poi linearly trasformate: $W(d) = \text{MLP}(\{e_k(d)\})$. SchNet è invariante a E(3): la distanza è invariante a rotazioni e traslazioni, e la somma sui vicini è invariante alla permutazione.

246.2 DimeNet: Interazioni Angolari

SchNet usa solo distanze — perde le informazioni angolari tra legami. DimeNet (Gasteiger et al., 2020) aggiunge le **angle embeddings** per ogni tripla (j, i, k) di atomi interagenti:

$$\mathbf{m}_{ji} = \text{Interaction}(\mathbf{m}_{ji}^{\text{prev}}, \{\mathbf{m}_{kj} : k \in \mathcal{N}(j) \setminus i\}, d_{ij}, \{\alpha_{kji} : k \in \mathcal{N}(j) \setminus i\})$$

Gli embeddings dell'angolo α_{kji} (tra i vettori $\mathbf{r}_j - \mathbf{r}_k$ e $\mathbf{r}_i - \mathbf{r}_j$) vengono espansi in funzioni di Bessel sferiche — la base naturale per descrivere la geometria 3D con invarianza di rotazione. DimeNet++ aggiunge downsampling per efficienza computazionale.

246.3 PaiNN: Equivarianza Vettoriale

PaiNN (Schütt et al., 2021) usa non solo feature scalari invarianti $\mathbf{h}_i \in \mathbb{R}^F$ ma anche feature vettoriali equivarianti $\mathbf{V}_i \in \mathbb{R}^{F \times 3}$ — vettori che ruotano con la molecola:

$$\Delta \mathbf{h}_i = \sum_j \phi_s(\mathbf{h}_j, d_{ij}), \qquad \Delta \mathbf{V}_i = \sum_j \phi_v(\mathbf{h}_j, d_{ij}) \cdot \hat{\mathbf{r}}_{ij}$$

dove $\hat{\mathbf{r}}_{ij} = (\mathbf{r}_j - \mathbf{r}_i)/d_{ij}$ è il vettore unitario interatomico. Le feature vettoriali possono essere usate per predire quantità vettoriali (forze atomiche, momenti di dipolo) o per costruire feature di ordine superiore (prodotti scalari tra vettori = scalari invarianti, prodotti vettoriali = pseudo-vettori equivarianti).

Modello	Input	Equivarianza	MAE energia (eV/atomo)	FLO
SchNet	Distanze	Invarianza E(3)	1.5	1×
DimeNet++	Distanze + angoli	Invarianza E(3)	0.3	3×
PaiNN	Dist. + vettori	Equivarianza E(3)	0.22	2×
EGNN (§77 Vol.II)	Distanze + coord.	Equivarianza E(3)	0.35	1.5×
NequIP (Batzner 2022)	Armoniche sferiche	Equivarianza E(3)	0.12	10×

Teoria dell'Informazione e Modelli di Linguaggio: Entropia, Cross-Entropia e Perplexity

247.1 Entropia di Shannon e Limite della Compressione

L'**entropia di Shannon** $H(X) = -\sum_x p(x) \log_2 p(x)$ (bit) o $-\mathbb{E}_p[\log p(x)]$ (nats) quantifica l'incertezza media di una variabile aleatoria. È il limite inferiore per la lunghezza media del codice ottimale (teorema di Shannon, 1948): nessun codice senza prefisso può avere lunghezza media inferiore a $H(X)$ bit.

La **cross-entropia** $H(p,q) = -\mathbb{E}_p[\log q(x)] = H(p) + D_{\mathrm{KL}}(p\|q)$ misura il costo medio di usare q come codice per dati distribuiti secondo p. Per i LLM, la training loss è la cross-entropia tra la distribuzione empirica dei dati p_{data} e le predizioni del modello q_θ:

$$\mathcal{L} = H(p_{\text{data}}, q_\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log q_\theta(x_i) \approx H(p_{\text{data}}) + D_{\mathrm{KL}}(p_{\text{data}}\|q_\theta)$$

Minimizzare la cross-entropia equivale a minimizzare la KL divergence tra dati e modello — l'entropia $H(p_{\text{data}})$ è fissa. La **perplexity** $\text{PPL} = e^{H(p,q)} = e^{\mathcal{L}}$ è la metrica standard per LLM: un modello con PPL k si comporta in media come una scelta uniforme tra k possibilità.

247.2 Mutual Information e Bottleneck Information

La **Mutual Information** $I(X;Y) = H(X) - H(X|Y) = D_{\mathrm{KL}}(p(x,y)\|p(x)p(y)) \geq 0$ misura quanto Y riduce l'incertezza su X. Proprietà: $I(X;Y) = I(Y;X)$; $I(X;Y) = 0$ iff $X \perp Y$; $I(X; f(X); Y) \leq I(X;Y)$ per qualsiasi f (data processing inequality).

L'**Information Bottleneck** (Tishby et al., 2000) formula l'apprendimento come un trade-off: trovare la rappresentazione T che massimizza $I(T;Y)$ (informazione sul target) e minimizza $I(T;X)$ (informazione sull'input — compressione):

$$\min_{p(t|x)} I(T;X) - \beta I(T;Y)$$

Il parametro β controlla il trade-off compressione-accuracy. La soluzione ottimale soddisfa l'equazione auto-consistente:

$$p(t|x) \propto p(t)\exp(-\beta D_{\mathrm{KL}}(p(y|x)\|p(y|t)))$$

$$p(t) = \sum_x p(t|x)p(x), \quad p(y|t) = \sum_x p(y|x)p(x|t)$$

Tishby e Schwartz-Ziv (2017) ipotizzano che le reti neurali implementino questo trade-off durante il training: prima memorizzano (alto $I(T;X)$, alto $I(T;Y)$), poi comprimono (basso

$I(T;X)$, $I(T;Y)$ mantenuto). La controversia è sulla misurabilità dell'MI nelle reti profonde — dipende fortemente dall'estimatore usato.

Neural Tangent Kernel: Reti Infinitamente Larghe e Convergenza

248.1 Il Neural Tangent Kernel

Il **Neural Tangent Kernel (NTK)** (Jacot et al., 2018) analizza il comportamento delle reti neurali nel limite di larghezza infinita. Per una rete $f(\mathbf{x};\boldsymbol{\theta})$, il NTK è:

$$\Theta(\mathbf{x},\mathbf{x}') = \nabla_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta})^T \nabla_{\boldsymbol{\theta}} f(\mathbf{x}';\boldsymbol{\theta})$$

Per reti di larghezza $n \to \infty$ con inizializzazione standard, $\Theta(\mathbf{x},\mathbf{x}')$ converge a un kernel deterministico $\Theta^*(\mathbf{x},\mathbf{x}')$ che *rimane costante durante il training* (il NTK è frozen). La dinamica di gradient flow con MSE loss diventa lineare:

$$\frac{d\mathbf{f}_t}{dt} = -\boldsymbol{\Theta}^*(\mathbf{X},\mathbf{X})(\mathbf{f}_t - \mathbf{y})$$

La soluzione esatta: $\mathbf{f}_t - \mathbf{y} = e^{-\boldsymbol{\Theta}^* t}(\mathbf{f}_0 - \mathbf{y})$. La convergenza è garantita se $\boldsymbol{\Theta}^*$ è positiva definita — il più piccolo autovalore $\lambda_{\min}(\boldsymbol{\Theta}^*) > 0$ determina la velocità di convergenza.

NTK Esplicito per MLP

Per un MLP a L layer con attivazione σ e larghezza $n_1 = \cdots = n_{L-1} = n \to \infty$, il NTK si calcola ricorsivamente. Definiamo la kernel ricorsione:

$$K^{(1)}(\mathbf{x},\mathbf{x}') = \frac{1}{d}\mathbf{x}^T\mathbf{x}' + \sigma_b^2$$

$$K^{(h+1)} = \sigma_w^2 \mathbb{E}_{(z,z')\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}^{(h)})}[\sigma(z)\sigma(z')] + \sigma_b^2$$

$$\text{dove } \boldsymbol{\Sigma}^{(h)} = \begin{pmatrix} K^{(h)}(\mathbf{x},\mathbf{x}) & K^{(h)}(\mathbf{x},\mathbf{x}') \\ K^{(h)}(\mathbf{x}',\mathbf{x}) & K^{(h)}(\mathbf{x}',\mathbf{x}') \end{pmatrix}$$

Il NTK finale per un MLP a L layer:

$$\Theta^{(L)}(\mathbf{x}, \mathbf{x}') = \sum_{h=1}^{L} \dot{K}^{(h)}(\mathbf{x}, \mathbf{x}') \prod_{h'=h+1}^{L} K^{(h')}(\mathbf{x}, \mathbf{x}')$$

$$\text{dove } \dot{K}^{(h)} = \sigma_w^2 \mathbb{E}_{(z,z')}[\dot{\sigma}(z)\dot{\sigma}(z')] + \sigma_b^2$$

dove $\dot{\sigma}$ è la derivata di σ. Il NTK è una somma di prodotti di kernel a ogni layer — cattura le interazioni a lungo raggio tra layer.

248.2 Limitazioni del NTK e Regime di Feature Learning

Il regime NTK descrive reti che non imparano feature — i pesi variano infinitesimamente dall'inizializzazione, quindi il modello è essenzialmente un kernel method. I modelli pratici però *imparano feature* — il regime di feature learning (μP, maximal update paramterization, Yang et al., 2022) descrive un'alternativa dove i pesi cambiano $O(1)$ e il NTK varia durante il training.

La distinzione pratica: con NTK scaling ($\sigma_w^2 = O(1/n)$), la rete scala male con la larghezza oltre un certo punto. Con μP scaling ($\sigma_w^2 = O(1/n)$ per layer intermedi, $O(1/n^2)$ per l'ultimo layer e $O(1)$ per il primo), la rete trasferisce gli iperparametri ottimali dalla piccola alla grande — lo stesso learning rate funziona su reti di larghezze diverse.

Ottimizzazione Non-Convessa: Selle, Minimi Locali e SGD

249.1 Geometria del Loss Landscape nelle Reti Profonde

Il loss landscape di una rete profonda ha milioni di dimensioni — l'intuizione geometrica è difficile ma alcune proprietà sono dimostrabili. Per reti overparametrizzate:

Teorema (Dauphin et al., 2014 — esteso)

Per reti sufficientemente large con random initialization: quasi tutti i punti critici (dove $\nabla \mathcal{L} = \mathbf{0}$) con loss alta sono selle saddle, non minimi locali. I minimi locali con alta loss sono esponenzialmente rari. La probabilità di un punto critico di essere un minimo locale con loss Δ sopra il minimo globale scala come $e^{-\Omega(n\Delta)}$ dove n è la larghezza.

La dimostrazione usa la connessione tra la ratio di autovalori negativi dell'Hessiana e l'indice della sella: per un punto con loss alta, la distribuzione degli autovalori dell'Hessiana tende ad avere molti negativi (la matrice di GOE — Gaussian Orthogonal Ensemble — ha $\approx 50\%$ autovalori negativi quando $\mathcal{L} > \mathcal{L}_{\min}$).

249.2 Saddle-Free Newton e Escape dalle Selle

Sulle selle, il gradiente è zero ma la Hessiana ha autovalori negativi. SGD con rumore riesce ad uscire dalle selle (il rumore "cade giù" lungo le direzioni negative), ma lentamente. Il **Saddle-Free Newton** usa il valore assoluto degli autovalori:

$$\Delta\boldsymbol{\theta} = -\eta|\mathbf{H}|^{-1}\nabla\mathcal{L}$$

dove $|\mathbf{H}|^{-1}$ sostituisce gli autovalori negativi con i loro assoluti — invece di salire lungo le direzioni concave, il metodo scende lungo tutte le direzioni. Il costo $O(|\theta|^3)$ lo rende inapplicabile, ma K-FAC approssimato può essere usato.

249.3 Sharpness-Aware Minimization: Analisi Teorica

SAM (§s83, Vol. II) trova minimi piatti minimizzando il worst-case loss in un intorno $\|\boldsymbol{\epsilon}\|_2 \leq \rho$:

$$\min_{\boldsymbol{\theta}} \max_{\|\boldsymbol{\epsilon}\| \leq \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$$

La soluzione del problema interno al primo ordine: $\boldsymbol{\epsilon}^* = \rho\nabla\mathcal{L}/\|\nabla\mathcal{L}\|_2$. L'aggiornamento SAM:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\nabla\mathcal{L}(\boldsymbol{\theta}_t + \rho\nabla\mathcal{L}_t/\|\nabla\mathcal{L}_t\|)$$

Connessione con PAC-Bayes: la sharpness-aware loss è correlata con il bound PAC-Bayes di McAllester (§133, Vol. III) con $Q = \mathcal{N}(\boldsymbol{\theta}, \rho^2\mathbf{I})$ e $P = \mathcal{N}(\mathbf{0}, \mathbf{I})$. SAM minimizza implicitamente il bound di generalizzazione PAC-Bayes — spiega teoricamente perché i minimi piatti generalizzano meglio.

249.4 Grokking: Generalizzazione Ritardata

Il **grokking** (Power et al., 2022) è un fenomeno in cui la rete raggiunge il 100% di training accuracy molto prima di raggiungere la generalizzazione — poi all'improvviso, dopo migliaia di epoche di overfit, generalizza perfettamente. Il fenomeno è stato osservato su task algoritmici (addizione modulare, permutazioni).

La spiegazione emergente: la rete prima memorizza il training set con circuiti specifici e efficienti (ma fragili). Solo con training prolungato i gradienti modificano lentamente i pesi verso circuiti più generalizzabili (algoritmi impliciti). Il peso della regolarizzazione (L2) accelera il grokking — penalizza i circuiti di memorizzazione che tendono ad avere pesi grandi.

Teoria della Misura per Reti Neurali: Wasserstein Gradient Flows

250.1 Il Problema della Particella come Ottimizzazione su Misure

La teoria classica dell'ottimizzazione opera in spazi euclidei finito-dimensionali: i parametri di una rete sono un vettore $\theta \in \mathbb{R}^p$ e il training è discesa del gradiente in questo spazio. Ma cosa succede quando la rete è così larga che i parametri di ogni singolo neurone sono meglio descritti come campioni da una *distribuzione*? Questa è la prospettiva del **regime di mean-field** (Mei, Montanari e Nguyen 2018; Chizat e Bach 2018), in cui si studia il limite di larghezza infinita e la distribuzione dei parametri di un singolo layer $\mu_t \in \mathcal{P}(\mathbb{R}^d)$ diventa l'oggetto di interesse matematico. In questo regime, la dinamica dell'addestramento può essere descritta in modo esatto come un **gradient flow nello spazio delle misure di probabilità**, dotato della metrica di Wasserstein W_2 — la distanza che misura il "costo di trasporto" tra distribuzioni. Questa connessione con la teoria del trasporto ottimale fornisce strumenti analitici potenti per capire convergenza e generalizzazione delle reti larghe.

Il funzionale di energia:

$$\mathcal{F}[\mu] = \mathbb{E}_{\mathbf{w}\sim\mu}[\mathcal{L}(\mathbf{w})] + \frac{\lambda}{2}\int\int K(\mathbf{w},\mathbf{w}')d\mu(\mathbf{w})d\mu(\mathbf{w}')$$

Il gradient flow di Wasserstein minimizza $\mathcal{F}$ rispetto alla metrica W_2:

$$\partial_t \mu_t = \nabla_{\mathbf{w}} \cdot (\mu_t \nabla_{\mathbf{w}} \frac{\delta \mathcal{F}}{\delta \mu_t})$$

dove $\delta\mathcal{F}/\delta\mu$ è la derivata funzionale. Questa è l'equazione di continuità per un fluido con velocità $\mathbf{v}(\mathbf{w}) = -\nabla_{\mathbf{w}}(\delta\mathcal{F}/\delta\mu)$ — i parametri si muovono come particelle in un campo di velocità. Per SGD: ogni parametro $\mathbf{w}_k$ è una "particella" e il training corrisponde all'evoluzione stocastica di questo sistema di particelle.

Universal Features e Polysemanticity nei Vision Transformer

251.1 Curve Detectors e High-Low Frequency Detectors

Olah et al. (2020) identificano famiglie di feature universali nelle reti convoluzionali — feature che appaiono indipendentemente nell'architettura e nel dataset di training. Le **universali feature di primo layer** includono: Gabor filters (frequenze orientate), color detectors (opponenza cromatica), frequency detectors (alta e bassa frequenza). Compaiono in InceptionNet, VGG, AlexNet — evidenza che le CNN convergono a feature simili indipendentemente dall'architettura.

I **curve detectors** al layer 3 si formano combinando Gabor filters orientati in modo da rispondere a curve di vario raggio. La geometria: ogni curve detector risponde a un arco di cerchio specifico, e il set completo di curve detectors copre uniformemente lo spazio degli archi. La composizione è verificabile: la feature "curva di 45° con raggio 20px" si ottiene dall'activation maximization del neurone e si decompone in contributi di Gabor filters orientati ad angoli appropriati.

251.2 Probing Classifiers e Sondaggio delle Rappresentazioni

I **probing classifiers** (Alain e Bengio, 2016; Belinkov et al., 2017) valutano quale informazione è lineare-estraibile da una rappresentazione intermedia. Dato un layer l che produce $\mathbf{h}^l \in \mathbb{R}^d$, si addestra un classificatore lineare su $\mathbf{h}^l$ per predire una proprietà linguistica (POS tag, constituency, profondità sintattica).

$$\text{Probe Accuracy}(l) = \max_{\mathbf{W},\mathbf{b}} \text{Acc}(\text{softmax}(\mathbf{W}\mathbf{h}^l + \mathbf{b}), \text{label})$$

La critica principale al probing (Hewitt e Liang, 2019): un probe altamente accurato può indicare che la proprietà è *estratta* dalla rappresentazione ma anche che il probe stesso è così espressivo da imparare la proprietà da qualsiasi rumore correlato. La soluzione: controllare la *selectivity* confrontando la probe accuracy su permutazioni random delle label — la selectivity $= \text{Acc}_{\text{probe}} - \text{Acc}_{\text{random}}$ misura la vera informazione.

Structural Probe per la Sintassi

Hewitt e Manning (2019) propongono il **structural probe**: invece di un classificatore lineare, si cerca una trasformazione lineare $\mathbf{B} \in \mathbb{R}^{k \times d}$ tale che le distanze nello spazio trasformato approssimino le distanze nell'albero sintattico:

$$d_{\mathbf{B}}(w_i, w_j) = \|\mathbf{B}(\mathbf{h}_i - \mathbf{h}_j)\|_2 \approx d_{\text{tree}}(w_i, w_j)$$

La dimensione k controlla la capacità del probe — con $k < d$, si forza la struttura sintattica a occupare un sottospazio. BERT-large con $k = 64$ raggiunge UUAS (unlabeled undirected attachment score) ≈ 0.87 — la struttura sintattica è decodificabile linearmente dalle sue rappresentazioni.

Mechanistic Anomaly Detection e Steering Vectors

252.1 Representation Engineering e Steering

Identificare direzioni nel residual stream di un LLM che corrispondono a concetti comportamentali — onestà, aggressività, seguire le istruzioni — e poi usare queste direzioni per modificare il comportamento del modello a inference time: questo è il programma della **representation engineering** (Zou, Phan, Chen, Campbell, Guo, Yin, Mazeika, Li, Song, Park e Hendrycks, 2023). A differenza del fine-tuning, che richiede gradient updates e rischia di degradare le capacità generali, la representation engineering interviene direttamente sulle attivazioni durante il forward pass, rendendo il controllo del comportamento immediato, reversibile e interpretabile. La representation engineering manipola le rappresentazioni interne del modello per controllare il comportamento. L'idea: alcune proprietà comportamentali del modello (onestà, seguire le istruzioni, emozioni) corrispondono a *direzioni* nel residual stream. Trovare queste direzioni permette di "sterzare" il modello.

Il metodo di estrazione dei **steering vectors**: si raccolgono coppie di prompt contrastive (x_+, x_-) dove x_+ elicita il comportamento target e x_- quello opposto. Si calcola la differenza media delle attivazioni al layer l:

$$\mathbf{v}_l = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{h}_l^{x_+^{(i)}} - \mathbf{h}_l^{x_-^{(i)}})$$

Si applica lo steering vector durante l'inferenza aggiungendolo al residual stream: $\mathbf{h}_l \leftarrow \mathbf{h}_l + \alpha \mathbf{v}_l$. Con $\alpha > 0$ si amplifica il comportamento target; con $\alpha < 0$ lo si sopprime. Zou et al. trovano direzioni per "onestà", "emotion", "refusal" — e verificano che manipolare queste direzioni cambia il comportamento come previsto.

252.2 Mechanistic Anomaly Detection: Trovare Backdoor

I meccanismi appresi dalla rete possono essere usati per rilevare **backdoor attack**: comportamenti nascosti attivati da trigger specifici. La detection meccanicistica (Goldowsky-Dill et al., 2023): si cerca un circuito nel computation graph che è attivo solo per gli input con trigger e produce un output anomalo.

Il metodo: si calcola la **activation difference** tra input puliti e input con trigger, e si usa il causal tracing per identificare i layer/testa responsabili del comportamento diverso. Un backdoor lascia un footprint nel computation graph — feature inusuali che si attivano solo per il pattern trigger.

Causal Discovery: Algoritmi PC, GES e LiNGAM

253.1 L'Algoritmo PC: Skeleton Learning + Orientamento

L'algoritmo **PC** (Peter-Clark, Spirtes et al., 2000) impara il CPDAG da dati osservazionali testando indipendenze condizionali. La fase di skeleton learning costruisce il grafo non-orientato:

Algoritmo: Algoritmo PC — Skeleton Learning

Input: variabili V, test di indipendenza $\perp$ con livello α

Init: grafo completo non-orientato su V

Per ogni coppia (X, Y) con arco e ogni $k = 0, 1, 2, \ldots$:

Per ogni $Z \subseteq \text{Adj}(X) \setminus Y$ con $|Z| = k$:

Se $X \perp Y | Z$: rimuovi arco $X - Y$, salva $\text{Sep}(X, Y) = Z$

Output: skeleton con $\text{Sep}(X, Y)$ per ogni coppia separata

La fase di orientamento: gli v-structures $X \to Z \leftarrow Y$ (collider) sono identificati se $Z \notin \text{Sep}(X, Y)$ (il collider NON d-separa). Gli archi restanti vengono orientati applicando le regole di Meek (1995) per evitare nuovi v-structure o cicli. L'algoritmo è corretto e completo sotto fedeltà e causalità Markov — converge al CPDAG del DAG generante.

253.2 LiNGAM: Identificazione Completa con Non-Gaussianità

I metodi basati su indipendenze condizionali identificano solo la Markov equivalence class — non il DAG completo. **LiNGAM** (Shimizu et al., 2006) sfrutta la non-gaussianità dei disturbi per identificare il DAG completo con modelli lineari:

$$\mathbf{x} = \mathbf{B}\mathbf{x} + \mathbf{e}, \quad \mathbf{e}_i \text{ indipendenti e non-gaussiani}$$

$$(\mathbf{I} - \mathbf{B})\mathbf{x} = \mathbf{e} \Rightarrow \mathbf{x} = (\mathbf{I} - \mathbf{B})^{-1}\mathbf{e} = \mathbf{A}\mathbf{e}$$

$\mathbf{A}$ è una matrice di mixing — questa è un'**ICA (Independent Component Analysis)** con la struttura addizionale che $\mathbf{A}^{-1} = (\mathbf{I} - \mathbf{B})$ è lower triangular (dopo permutazione delle variabili in ordine causale). DaphillHyvarinen (2004) provano: per disturbi non-gaussiani, l'ICA identifica $\mathbf{A}$ fino a permutazione e scala. Il teorema di identificazione LiNGAM: il DAG causale è identificabile dalla distribuzione osservazionale se i disturbi sono non-gaussiani.

253.3 Counterfactual Data Augmentation per Robustezza

La robustezza fuori distribuzione può essere migliorata con **counterfactual data augmentation (CoDA)**: si generano esempi controfattuali variando solo le feature causalmente rilevanti per il target. Invece di addestrare su $p(x, y)$, si addestra su $p(x, y) + p(x_{\text{cf}}, y_{\text{cf}})$ dove x_{cf} è la versione di x con le correlazioni spurie rimosse.

L'implementazione con LLM: dato un testo x con label y (sentimento "positivo" su una recensione di film), si chiede a un LLM di generare x_{cf} cambiando solo gli elementi spuriamente correlati (contesto cinematografico) mantenendo il sentimento. Il modello addestrato con CoDA generalizza meglio su test set out-of-domain.

Architectures vs Scaling: MoE, State Space Models e Confronto

254.1 Mixture of Experts vs Dense: Il Confronto Teorico

I MoE (§161, Vol. III) offrono un trade-off diverso rispetto ai modelli densi: a parità di FLOPs di inferenza, i MoE hanno più parametri totali e quindi più capacità. La comparazione rigorosa richiede di fissare una variabile — FLOPs, parametri, o compute di training.

Per FLOPs di training fissi (C), la legge di scaling MoE:

$$L_{\text{MoE}}(N_{\text{total}}, E, k) = L_{\text{dense}}(N_{\text{active}}) + \Delta_{\text{routing}}$$

dove $N_{\text{active}} = N_{\text{total}}/E \cdot k$ è il numero di parametri attivi per token, e Δ_{routing} è il costo del load imbalance. In pratica: $E = 8$, $k = 2$ (Mixtral) fa sì che il modello usi $2/8 = 25\%$ dei parametri per ogni token — a parità di FLOPs, si "vedono" $8\times$ più parametri durante il training. Empiricamente, MoE con 8 expert e top-2 raggiunge le stesse performance di un modello denso con $\sim 2\times$ i FLOPs di training risparmiati.

254.2 State Space Models (Mamba) vs Attention: Complessità e Qualità

Mamba (Gu e Dao, 2023) propone un'alternativa all'attention basata su State Space Models (SSM) selettivi. L'SSM standard è un sistema lineare time-invariant:

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t), \quad y(t) = \mathbf{C}h(t)$$

Discretizzato con step size Δ: $\bar{\mathbf{A}} = e^{\Delta \mathbf{A}}$, $\bar{\mathbf{B}} = (\Delta \mathbf{A})^{-1}(e^{\Delta \mathbf{A}} - \mathbf{I})\Delta \mathbf{B}$. L'output:

$$h_t = \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t, \quad y_t = \mathbf{C}h_t$$

La chiave di Mamba: i parametri $\mathbf{B}$, $\mathbf{C}$, Δ diventano funzioni dell'input x_t (selective SSM) — la rete decide quanto "ricordare" a ogni step. Il training usa la convoluzione parallelizzabile; l'inferenza usa la ricorrenza $O(1)$ per token. Per sequenze lunghe N: training $O(N \log N)$ (FFT-based convolution), inferenza $O(1)$ per step.

Confronto con attention: la qualità di Mamba su compiti di recall (copiare un valore visto molti step prima) è inferiore all'attention — l'hidden state finito non può memorizzare tutto. Ma

su compiti di comprensione linguistica generale, Mamba-3B compete con Transformer-3B su sequenze di lunghezza moderata.

Quantizzazione Post-Training e Calibrazione: GPTQ, AWQ e SmoothQuant

255.1 GPTQ: Optimal Brain Compression

GPTQ (Frantar et al., 2023) quantizza ogni layer di un LLM risolvendo il problema di ottimizzazione layer-wise. Per un layer lineare $\hat{\mathbf{W}} = \arg\min_{\mathbf{Q}} \|(\mathbf{W} - \mathbf{Q})\mathbf{X}\|_F^2$ dove $\mathbf{Q}$ è la matrice quantizzata, si può riscrivere come:

$$E = \|(\mathbf{W} - \mathbf{Q})\mathbf{X}\|_F^2 = \text{tr}((\mathbf{W} - \mathbf{Q})\mathbf{H}(\mathbf{W} - \mathbf{Q})^T)$$

dove $\mathbf{H} = \mathbf{X}\mathbf{X}^T$ è la Gram matrix. GPTQ quantizza sequenzialmente colonna per colonna, propagando l'errore alle colonne rimanenti. Per la colonna q già quantizzata, l'errore ottimale propagato alle colonne $j > q$:

$$\delta_j = -\frac{w_{q,j}^{\text{err}}}{[\mathbf{H}^{-1}]_{qq}} \cdot [\mathbf{H}^{-1}]_{q,j:}$$

L'inversione di $\mathbf{H}$ si fa una volta con Cholesky decomposition, poi gli aggiornamenti sono $O(d^2)$ per colonna. GPTQ quantizza LLaMA-65B a INT4 in ~ 4 ore su una singola A100, con perdita di perplexity < 1 punto rispetto al modello FP16.

255.2 AWQ: Activation-Aware Weight Quantization

AWQ (Lin et al., 2023) osserva che non tutti i pesi sono ugualmente importanti — alcuni canali hanno attivazioni molto più grandi e contribuiscono maggiormente all'output. La quantizzazione uniforme perde più informazione sui canali "importanti".

La soluzione: scalare i pesi prima della quantizzazione per ridurre l'errore sui canali importanti. Per il canale i con scale s_i:

$$\mathbf{W}' = \mathbf{W} \cdot \text{diag}(\mathbf{s}), \quad \mathbf{X}' = \text{diag}(\mathbf{s})^{-1} \cdot \mathbf{X}$$

Il prodotto $\mathbf{W}'\mathbf{X}' = \mathbf{W}\mathbf{X}$ è invariante — si scala solo prima di quantizzare i pesi. La scale s_i ottimale per ogni canale si trova minimizzando l'errore di quantizzazione su un piccolo calibration set: $\mathbf{s}^* = \arg\min_{\mathbf{s}} \|Q(\mathbf{W} \cdot \mathbf{s}) \cdot \mathbf{s}^{-1} \cdot \mathbf{X} - \mathbf{W}\mathbf{X}\|$. AWQ raggiunge qualità simile a GPTQ con un metodo più semplice e veloce.

Speculative Decoding: Derivazione e Analisi della Velocità

256.1 Il Meccanismo dello Speculative Decoding

Lo speculative decoding (Leviathan et al., 2023; Chen et al., 2023) rompe questa serialità con un'idea controintuitiva: non è necessario che ogni token del draft sia corretto — basta che il modello target possa *verificare* i token proposti in parallelo, molto più velocemente che generarli uno per uno. Il sistema usa un modello "draft" M_q piccolo e veloce per proporre K token, e il modello target M_p grande per verificarli in parallelo.

Algoritmo: Speculative Decoding — Un Ciclo

1. Draft: M_q genera K token autoregressivamente $\rightarrow$ sequenza $\tilde{x}_1, \ldots, \tilde{x}_K$

2. Verify: M_p calcola in un *singolo* forward pass tutte le distribuzioni $p(x_t|\text{context}, x_{1:t-1})$

3. Accept/Reject: per ogni token $\tilde{x}_t$ proposto:

$r \sim \text{Uniform}[0, 1]$

Se $r \leq p(\tilde{x}_t)/q(\tilde{x}_t)$: *accetta*

Altrimenti: *rigetta, campiona dalla distribuzione corretta residua* $p'(x) = \max(0, p - q)/Z$

4. Avanza il contesto fino all'ultimo token accettato

La procedura di accettazione/rifiuto è progettata per produrre la distribuzione *esatta* di M_p — lo speculative decoding non è un'approssimazione. Il token finale (dopo il primo rifietto o dopo K accettazioni) viene sempre campionato da M_p per garantire la correttezza.

256.2 Analisi del Throughput

Sia $\alpha = \mathbb{E}_t[p(\tilde{x}_t \leq q(\tilde{x}_t))]$ il tasso medio di accettazione. Il numero atteso di token accettati per forward pass di M_p:

$$\mathbb{E}[\text{tokens per step}] = \frac{1 - \alpha^{K+1}}{1 - \alpha}$$

Il speedup rispetto alla generazione standard (1 token per forward pass di M_p) dipende dalla velocità relativa di M_q e dalla acceptance rate α. Sia c il costo relativo di M_q rispetto a M_p: uno speedup di $s\times$ richiede $\alpha > (s(1 + Kc) - 1)/(s - 1)$. Per $c = 0.1$ (modello draft $10\times$ più piccolo), $K = 5$, e $\alpha = 0.8$: speedup $\approx 2.5\times$ — confermato empiricamente con drafts come LLaMA-68M per LLaMA-13B.

Privacy Amplificazione: Shuffling e Subsampling

257.1 Amplificazione tramite Subsampling

Se il meccanismo $\mathcal{M}$ è $(\varepsilon_0, \delta_0)$-DP e viene applicato solo a un subset di m record scelti uniformemente da n record (con sampling rate $q = m/n$), il meccanismo composto è "amplificato":

Teorema — Privacy Amplification by Subsampling

$$\mathcal{M}_{\text{subsampled}} \text{ è } (O(q\varepsilon_0), q\delta_0 + \delta')\text{-DP}$$

Più precisamente: $(\log(1 + q(e^{\varepsilon_0} - 1)), q\delta_0)$-DP. Per ε_0 piccolo: $\approx (q\varepsilon_0, q\delta_0)$-DP — il budget si moltiplica per q.

Intuizione: un avversario che osserva l'output non sa se il proprio record è stato incluso nel subset — la sua influenza è diluita dalla probabilità di sampling. L'amplificazione è "gratuita" — la stessa implementazione del meccanismo diventa più privata se applicata su minibatch.

257.2 Amplificazione tramite Shuffling

Il **modello shuffle** (Erlingsson et al., 2019; Cheu et al., 2019) aggiunge un terzo attore tra i client e il server: un shuffler fidato che riceve i messaggi privati dai client e li permuta casualmente prima di inviarli al server. Il server vede solo i messaggi permutati — non sa quale client ha inviato quale messaggio.

Amplificazione tramite Shuffling (Feldman et al., 2021)

Se ogni client applica un meccanismo ε_0-LDP (local DP) sui propri dati e uno shuffler permuta i n messaggi, il meccanismo aggregato è $O(\varepsilon_0 e^{\varepsilon_0}\sqrt{\log(1/\delta)/n}, \delta)$-DP. Per $\varepsilon_0 = O(1)$: amplificazione di $\sim \sqrt{n}$ rispetto alla LDP pura.

La garanzia è molto più forte della LDP (ε_0 per ogni record) e quasi equivalente al meccanismo centralizzato ($\varepsilon_0/\sqrt{n}$). Il modello shuffle è interessante per federated learning: i client hanno ε_0-LDP (nessuno vede i loro dati grezzi), ma il server ottiene una garanzia quasi-centralizzata senza fidarsi dei client.

Machine Unlearning: Dimenticare Dati Senza Re-Training

258.1 Il Problema dell'Unlearning

Il **machine unlearning** è il problema di rimuovere l'influenza di specifici dati di training da un modello già addestrato — senza re-training completo. Motivazione: GDPR "right to be forgotten", dati di training tossici, dati con copyright.

La definizione formale: dato un modello $\mathcal{M}$ addestrato su D e un subset $D_f \subset D$ da dimenticare, si vuole un modello $\mathcal{M}'$ tale che $\mathcal{M}'$ sia indistinguibile da un modello $\mathcal{M}_r$ addestrato su $D \setminus D_f$ (il "retrained" model). La distanza tra le distribuzioni output:

$$\|\mathcal{M}'(\cdot) - \mathcal{M}_r(\cdot)\|_{\text{TV}} < \varepsilon_{\text{forget}}$$

Il re-training completo garantisce l'unlearning ma ha costo $O(\text{training cost})$. I metodi efficienti approssimano l'unlearning con costo $O(\text{forgetting data})$.

258.2 Gradient Ascent e SCRUB

Il metodo più semplice: **gradient ascent** sui dati da dimenticare — invece di minimizzare la loss su D_f, la si *massimizza*, spingendo il modello verso la predizione casuale. Il problema: gradient ascent non-regolarizzato può degradare la performance sui dati da mantenere D_r.

SCRUB (Kurmanji et al., 2023) bilancia il dimenticare e il mantenere:

$$\mathcal{L}_{\text{SCRUB}} = \underbrace{-\lambda \mathbb{E}_{D_f}[\mathcal{L}(\mathcal{M}, x)]}_{\text{forget}} + \underbrace{\mathbb{E}_{D_r}[D_{\text{KL}}(\mathcal{M}(x) \| \mathcal{M}_0(x))]}_{\text{retain}}$$

Il secondo termine (KL divergence rispetto al modello originale $\mathcal{M}_0$) forza il modello a mantenere il comportamento sui dati da conservare — evita la "catastrophic forgetting" dei dati buoni.

Verifica dell'Unlearning: Membership Inference come Test

Si usa il MIA (§229) per verificare se i dati dimenticati sono stati effettivamente rimossi: se $\mathcal{M}'$ ha subito unlearning corretto, il LiRA score su D_f deve essere ≈ 0.5 (indistinguibile da un modello che non ha mai visto D_f). In pratica, nessun metodo di unlearning raggiunge garanzie forti su modelli grandi — rimane un problema aperto.

Voice Conversion, Speaker Verification e Diarization

259.1 Speaker Embedding: d-vectors e x-vectors

I sistemi di **speaker verification** devono decidere se due audio appartengono allo stesso parlante. La pipeline moderna estrae un **speaker embedding** (vettore compatto che cattura l'identità del parlante) e poi confronta i cosine similarity.

Gli **x-vectors** (Snyder et al., 2018: un TDNN (Time Delay Neural Network) con pooling statistico. La x-vector architecture:

- 5 layer TDNN — operano su diversi contesti temporali $\{t-2, t-1, t, t+1, t+2\}$, ecc.
- **Statistics pooling**: aggrega le attivazioni di tutti i frame T in media e std: $\mathbf{h}_{\text{pool}} = [\frac{1}{T}\sum_t \mathbf{h}_t; \sqrt{\frac{1}{T}\sum_t (\mathbf{h}_t - \bar{\mathbf{h}})^2}]$
- 2 layer fully-connected — l'output del primo è la x-vector

Training con softmax su speaker ID + addestramento discriminativo con **Angular Margin Loss (ArcFace)**:

$$\mathcal{L}_{\text{ArcFace}} = -\frac{1}{N}\sum_i \log \frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))} + \sum_{j\neq y_i} e^{s\cos\theta_j}}$$

Il margine angolare m (tipicamente $m = 0.5$ rad) forza i campioni dello stesso parlante ad essere più vicini tra loro di m radianti, rendendo gli embedding più discriminativi. Durante l'inferenza si usa cosine similarity: score $= \cos(\theta(\mathbf{e}_1, \mathbf{e}_2))$, con soglia calibrata su un dev set.

259.2 Voice Conversion con Normalizing Flows

La **voice conversion** trasforma la voce di un parlante sorgente in quella di un target, mantenendo il contenuto linguistico. L'approccio basato su normalizing flows (Casanova et al., 2022):

$$\mathbf{z}_{\text{content}} = f_{\text{content}}(\mathbf{S}) \qquad \text{(flow invertibile, contenuto)}$$

$$\mathbf{z}_{\text{speaker}} = f_{\text{speaker}}(\text{reference audio}) \qquad \text{(speaker embedding)}$$

$$\mathbf{S}_{\text{converted}} = f^{-1}_{\text{content}}(\mathbf{z}_{\text{content}}; \mathbf{z}_{\text{speaker}}) \qquad \text{(sintesi con nuovo speaker)}$$

Il content flow impara una rappresentazione content-independent invertibile. La speaker embedding viene iniettata come condizionamento nella sintesi. L'invertibilità del flow garantisce che nessuna informazione di contenuto vada persa nella conversione.

Source Separation e Beamforming Neurale

260.1 Conv-TasNet: Separazione nel Dominio del Tempo

Il problema della separazione delle sorgenti audio — estrarre voci o strumenti individuali da un mix — ha tradizionalmente richiesto una rappresentazione tempo-frequenza come lo spettrogramma di magnitudine STFT, su cui si applicano maschere. Questa scelta introduce tuttavia un disaccoppiamento indesiderato: la fase del segnale viene tipicamente ignorata o ricostruita euristicamente, e la risoluzione tempo-frequenza del STFT richiede un compromesso fisso tra localizzazione temporale e spettrale. **Conv-TasNet** (Luo e Mesgarani, 2019) propone un approccio radicalmente diverso: separare le sorgenti direttamente nel dominio del tempo, imparando end-to-end sia la rappresentazione del segnale che le maschere di separazione, eliminando la necessità di qualsiasi trasformata a mano. Il risultato è una pipeline completamente differenziabile che supera significativamente i metodi basati su STFT su benchmark standard come WSJ0-2mix. L'architettura usa un encoder convoluzionale 1D non-lineare, una TCN (Temporal Convolutional Network) per stimare le maschere, e un decoder:

$$\mathbf{w} = \text{Encoder}(\mathbf{x}) = \text{ReLU}(\mathbf{U} \cdot \mathbf{x}) \in \mathbb{R}^{N \times T}$$

$$\mathbf{m}_k = \text{Sigmoid}(\text{TCN}_k(\mathbf{w})) \in [0,1]^{N \times T} \quad \text{per sorgente } k$$

$$\hat{\mathbf{x}}_k = \text{Decoder}(\mathbf{m}_k \odot \mathbf{w}) = (\mathbf{m}_k \odot \mathbf{w})^T \mathbf{V}$$

Il training minimizza la **Scale-Invariant SDR (SI-SDR)**:

$$\text{SI-SDR}(\mathbf{s}, \hat{\mathbf{s}}) = 10 \log_{10} \frac{\| \frac{\hat{\mathbf{s}}^T \mathbf{s}}{\|\mathbf{s}\|^2} \mathbf{s} \|^2}{\| \hat{\mathbf{s}} - \frac{\hat{\mathbf{s}}^T \mathbf{s}}{\|\mathbf{s}\|^2} \mathbf{s} \|^2}$$

La loss è invariante alla scala — evita soluzioni degeneri dove il modello amplifica arbitrariamente il segnale. Il problema di **permutation invariance**: con K sorgenti, ci sono $K!$ modi di assegnare le output alle sorgenti target. Si risolve con Permutation Invariant Training (PIT): si ottimizza su tutti i $K!$ assegnamenti e si usa il migliore.

Reward Learning da Preferenze Umane per Robotica

261.1 RLHF per Task Robotici

Definire funzioni di reward per task robotici complessi è uno dei problemi più ostici del reinforcement learning applicato: una reward mal specificata produce comportamenti inattesi o soluzioni che soddisfano la lettera ma non lo spirito del compito (reward hacking). L'alternativa proposta da Christiano et al. (2017) è elegante: è più facile per un operatore umano mostrare quale tra due traiettorie è migliore che specificare una funzione di reward matematica. Il reward learning da preferenze impara implicitamente la reward function dall'aggregazione di tali giudizi comparativi. Il **reward learning da preferenze** (Christiano et al., 2017 per RL; Biyik et al., 2022 per robotica) addestra un reward model da confronti umani tra traiettorie.

Dati confronti $\{(\tau_1^{(i)}, \tau_2^{(i)}, y^{(i)})\}$ dove $y = 0$ se τ_1 è preferita, $y = 1$ altrimenti, si stima il reward $r_\psi(\tau)$ minimizzando la Bradley-Terry loss (come in RLHF, §116b Vol. III):

$$\mathcal{L}_{\text{reward}} = -\sum_i [y^{(i)} \log P(\tau_2^{(i)} \succ \tau_1^{(i)}) + (1 - y^{(i)}) \log P(\tau_1^{(i)} \succ \tau_2^{(i)})]$$

dove $P(\tau_1 \succ \tau_2) = \sigma(R_\psi(\tau_1) - R_\psi(\tau_2))$ con $R_\psi(\tau) = \sum_t r_\psi(s_t, a_t)$. Il reward model apprende dal feedback umano e guida l'ottimizzazione della policy con PPO.

Active Learning per la Riduzione del Feedback

L'active learning riduce il numero di confronti richiesti selezionando le coppie di traiettorie più informative. La metrica di informatività (information gain del reward model):

$$\text{IG}(\tau_1, \tau_2) = H(P(y|\tau_1, \tau_2)) - \mathbb{E}_{\psi \sim P(\psi|\mathcal{D})}[H(P(y|\tau_1, \tau_2, \psi))]$$

In pratica, si usa un ensemble di reward models $\{r_{\psi_k}\}$ e si seleziona la coppia con la più alta varianza tra i reward predetti — massimizza il disaccordo tra i modelli e quindi l'informazione guadagnata dal feedback. Con active learning, si può raggiungere la stessa qualità con 10× meno confronti umani.

Sim-to-Real Transfer: Domain Randomization e Adaptation

262.1 Domain Randomization

La simulazione fisica offre un'opportunità straordinaria per il RL: dati di training illimitati, reset istantanei, parallelizzazione massiva — in pochi giorni si possono raccogliere miliardi di transizioni che richiederebbero anni nel mondo fisico. Ma questa abbondanza di dati simulati ha un tallone d'Achille: le reti addestrate solo in simulazione spesso falliscono nel mondo reale a causa del **reality gap** — differenze di fisica, rendering, attriti, rumori. La **domain randomization** (Tobin et al., 2017) addestra la policy su una distribuzione di simulazioni con parametri fisici variabili, rendendo la policy robusta a queste variazioni:

$$\pi^* = \arg\max_\pi \mathbb{E}_{\phi \sim p(\phi)}[J(\pi, \text{Sim}_\phi)]$$

I parametri randomizzati ϕ includono: massa degli oggetti, coefficienti di attrito, rigidità dei giunti, latenza dell'attuatore, livelli di luce e texture delle superfici. Se la distribuzione reale

$p_{\text{real}}(\phi)$ è contenuta nella distribuzione di randomizzazione $p(\phi)$, la policy addestrata generalizza al mondo reale.

262.2 Adversarial Domain Randomization

La randomizzazione uniforme non è ottimale — alcuni parametri sono più critici per il task. L'**ADR (Adversarial Domain Randomization)** (Akkaya et al., 2019, OpenAI Five per Rubik's Cube) adatta la distribuzione durante il training: aumenta la varianza dei parametri per cui la policy è già robusta, e la riduce per quelli che causano fallimenti frequenti.

La regola di aggiornamento della distribuzione $p(\phi_i) = \text{Uniform}[\phi_i^{\text{lo}}, \phi_i^{\text{hi}}]$: se la performance media supera la soglia τ_{hi}, si espande il range; se scende sotto τ_{lo}, si contrae:

$$\phi_i^{\text{hi}} \leftarrow \phi_i^{\text{hi}} + \delta \text{ se } \bar{R} > \tau_{\text{hi}}, \qquad \phi_i^{\text{hi}} \leftarrow \phi_i^{\text{hi}} - \delta \text{ se } \bar{R} < \tau_{\text{lo}}$$

OpenAI ha usato ADR per risolvere il cubo di Rubik con una mano robotica — la distribuzione finale di randomizzazione copriva variazioni fisiche che includevano oggetti mai visti in training (gusci d'uovo attaccati al cubo, guanti da ginnastica sul robot).

RFdiffusion e Proteine de Novo Design

263.1 Diffusione nello Spazio delle Strutture Proteiche

Il protein design computazionale — progettare sequenze aminoacidiche che si ripieghino in strutture 3D con funzioni predeterminate — è uno dei problemi più ambiziosi della biologia computazionale. Per decenni è stato affrontato con metodi di ottimizzazione energetica basati su campi di forza fisici (Rosetta), ma questi metodi richiedono molte ore di calcolo e faticano a esplorare lo spazio conformazionale in modo globale. L'avvento di AlphaFold2 per la predizione della struttura ha aperto la strada all'uso di reti neurali anche per il problema inverso del design. **RFdiffusion** (Watson et al., 2022, Baker Lab) porta il paradigma dei modelli di diffusione direttamente nello spazio delle strutture proteiche — genera strutture 3D de novo condizionatamente su specifici motivi funzionali, come siti di binding o motivi di coordinazione di metalli, senza bisogno di un template di partenza. La variabile di diffusione è la tupla di frame rigidi $(R_i, \mathbf{t}_i)$ per ogni residuo.

Il processo forward rumorisca i frame applicando perturbazioni SE(3): rotazioni casuali da IGSO(3) (Isotropic Gaussian SO(3)) e traslazioni gaussiane:

$$R_i^{(t)} = R_i^{(0)} \cdot \exp(\sigma_t \hat{\omega}), \quad \mathbf{t}_i^{(t)} = \mathbf{t}_i^{(0)} + \sigma_t \boldsymbol{\varepsilon}$$

dove $\hat{\omega} \sim \text{IGSO}(3)$ è una rotazione aleatoria isotropa e $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. La rete di denoising è RoseTTAFold (simile a AlphaFold2) che predice i frame "denoisati" dato il frame corrente e il livello di rumore t.

Il condizionamento: si può specificare un motivo strutturale (sito attivo, epitopo) che deve essere presente nella struttura generata — la diffusione genera la struttura che "circonda" il motivo. RFdiffusion genera proteine con specificità di binding verso target predefiniti, accelerando il drug discovery.

Single-Cell RNA Sequencing e Foundation Models per Biologia

264.1 scRNA-seq come Sequenza di Geni

Il **single-cell RNA sequencing (scRNA-seq)** misura l'espressione di migliaia di geni in ogni singola cellula — produco matrici di espressione $\mathbf{X} \in \mathbb{R}^{N_{\text{cells}} \times G_{\text{genes}}}$ dove X_{cg} è il numero di trascritti del gene g nella cellula c. I valori sono interi (count data) con distribuzione negativa binomiale.

Geneformer (Theodoris et al., 2023): le cellule vengono "tokenizzate" come sequenze di geni ordinati per espressione (geni più espressi prima). Un BERT pre-addestrato su 30M cellule predice i geni mascherati — impara le relazioni funzionali tra geni. Il masked gene prediction objective:

$$\mathcal{L} = -\sum_{g \in \mathcal{M}} \log p(\text{gene}_g | \text{cell context})$$

dove $\mathcal{M}$ sono i geni mascherati. Geneformer raggiunge stato dell'arte su predizione di perturbazioni genetiche, identificazione di tipi cellulari rari, e predizione dell'effetto di drug su cell types specifici — zero-shot su task non visti durante il pretraining.

264.2 Variational Autoencoders per Biologia: scVI

scVI (Lopez et al., 2018: un VAE per dati scRNA-seq che modella esplicitamente le fonti di variazione biologica e tecnica. Il generative model:

$$z_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \ell_n \sim \text{LogNormal}(\ell_{\mu_n}, \ell_{\sigma_n^2})$$

$$\rho_{ng} = f_\theta(z_n, s_n) \qquad \text{(proporzione gene-specifiche)}$$

$$x_{ng} \sim \text{NegativeBinomial}(\ell_n \rho_{ng}, \theta_g^{-1})$$

dove s_n è il batch indicator (diversi esperimenti, diverse piattaforme), ℓ_n è il library size (numero totale di trascritti), e θ_g è il parametro di dispersione per gene. Il VAE impara a separare la variazione biologica (z_n, latente) dalla variazione tecnica (ℓ_n, s_n, modellata esplicitamente). Il latent space z_n può essere usato per clustering dei tipi cellulari e per rimozione dei batch effects.

Predizione del Legame Proteina-Molecola: Docking Differenziabile

265.1 Il Problema del Molecular Docking

La scoperta di nuovi farmaci è storicamente un processo empirico costoso: sintetizzare e testare sperimentalmente milioni di composti candidati richiede anni e miliardi di dollari. Il **docking molecolare computazionale** mira a virtualizzare questo screening: predice come una piccola molecola (ligando) si lega a una proteina bersaglio (recettore) e con quale affinità — informazioni essenziali per drug discovery. Il problema è fondamentalmente un'ottimizzazione nello spazio delle pose 3D del ligando (posizione, orientazione, conformazione dei legami rotabili), con una funzione obiettivo (scoring function) che approssima l'energia di legame libera. I metodi classici come AutoDock Vina usano funzioni di scoring fisicamente motivate e ricerca euristica; i metodi neurali differenziabili trattano l'intero problema come ottimizzazione differenziabile. Il problema: trovare la posa (posizione + conformazione) del ligando che minimizza l'energia di binding:

$$\text{posa}^* = \arg\min_{(\mathbf{R},\mathbf{t},\boldsymbol{\chi})} \mathcal{E}(\text{ligando}(\mathbf{R}, \mathbf{t}, \boldsymbol{\chi}), \text{proteina})$$

dove $\boldsymbol{\chi}$ sono gli angoli diedri (conformazione flessibile). I metodi classici (Glide, AutoDock) usano funzioni di scoring empiriche e ricerca stocastica (Monte Carlo, genetici). I metodi neurali imparano la scoring function e/o la generazione della posa.

DiffDock: Docking come Diffusione

DiffDock (Corso et al., 2023: un modello di diffusione su SE(3)$0148\mathbf{t}_t^*, \mathbf{R}_t^*, \boldsymbol{\chi}_t^* = \text{EGNN}_\theta(\text{proteina, ligando,}$ $\min_{\pi_2} \max_{\pi_1} \mathbb{E}[u_1] DISPMATH0150 \hat{\pi}_k^{(t)} = \frac{1}{t} \sum_{s=1}^{t} \mathbf{1}[a_k^{(s)}], \qquad a_k^{(t+1)} = \text{BR}(\hat{\pi}_{-k}^{(t)}) DISPMATH0151 Q_{\text{tot}}(\boldsymbol{\tau},$ $f_\phi(Q_1(\tau_1, a_1), \ldots, Q_n(\tau_n, a_n), s) DISPMATH0152 m_i^{(t)} = \text{Encoder}_i(o_i^{(t)}, h_i^{(t-1)}) DISPMATH0153 a_i^{(t)}, h_i^{(t+1)}$ $\text{DRU}_i(o_i^{(t)}, \{m_j^{(t)}\}_{j \neq i}, h_i^{(t)}) DISPMATH0154 I = \frac{\text{FLOPs}}{\text{bytes letti+scritti}} \quad (\text{FLOPs/byte}) DISPMATH0155 P_{\max} =$ $\min(\Pi, \beta \cdot I)$ FLOPs/sec $DISPMATH0156 \underbrace{\text{SeqParallel}}_{\text{LayerNorm}} \xrightarrow{\text{AllGather}} \underbrace{\text{TensorParallel}}_{\text{MLP/Attn}} \xrightarrow{\text{ReduceScatter}}$

$\underbrace{\text{SeqParallel}}_{\text{Dropout+LayerNorm}} \quad DISPMATH0157 v_{t+1}(\mathbf{x}) = \sigma(\mathbf{W} v_t(\mathbf{x}) + \mathcal{F}^{-1}[\mathbf{R}_\phi \cdot \mathcal{F}[v_t]](\mathbf{x})) \, DISPMATH0158 \mathcal{G}[u](y) \approx$ $\sum_{k=1}^{p} \underbrace{\text{BranchNet}(u(\mathbf{x}_1, \ldots, u(\mathbf{x}_m)))_k}_{\text{coefficienti}} \cdot \underbrace{\text{TrunkNet}(y)_k}_{\text{basis function}} DISPMATH0159 \mathcal{W}_p^p(\mu, \nu) = \inf_{\gamma \in \Pi(\mu,\nu)} \int_{\mathcal{X} \times \mathcal{Y}} c(\mathbf{x}, \mathbf{y})^p$ $\sup_{\|f\|_L \leq 1} \left[\int f d\mu - \int f d\nu \right] DISPMATH0161 \mathcal{OT}_\varepsilon(\mu, \nu) = \min_{\gamma \in \Pi(\mu,\nu)} \langle \mathbf{C}, \gamma \rangle - \varepsilon H(\gamma) DISPMATH0162 \mathbf{u}^{(t}$ $\mu / (\mathbf{K}\mathbf{v}^{(t)}), \qquad \mathbf{v}^{(t+1)} = \nu / (\mathbf{K}^T \mathbf{u}^{(t+1)}) DISPMATH0163 \bar{\mathbf{W}} = \mathbf{W} / \sigma(\mathbf{W}), \quad \sigma(\mathbf{W}) = \max_{\|\mathbf{u}\|=1} \|\mathbf{W}\mathbf{u}\| =$ $\sigma_1(\mathbf{W}) DISPMATH0164 \boldsymbol{\phi}(\mathbf{x}) = \sqrt{2/D}[\cos(\boldsymbol{\omega}_1^T \mathbf{x} + b_1), \ldots, \cos(\boldsymbol{\omega}_D^T \mathbf{x} + b_D)] DISPMATH0165 \lambda^*(t) =$ $\lim_{\Delta t \to 0} \frac{\mathbb{P}(\text{evento in } [t, t+\Delta t] | \mathcal{H}_t)}{\Delta t} DISPMATH0166 \log p(\{t_1, \ldots, t_N\}) = \sum_{i=1}^{N} \log \lambda^*(t_i) - \int_0^T \lambda^*(t) dt DISPMAT$ $\mu + \sum_{t_i < t} \phi(t - t_i) = \mu + \sum_{t_i < t} \alpha e^{-\beta(t-t_i)} DISPMATH0168 c(t) = c_{t_i}^{target} + (c_{t_i}^{cell} - c_{t_i}^{target}) e^{-\delta_{t_i}(t-t_i)}, \quad t \in$ $(t_i, t_{i+1}) DISPMATH0169 \lambda_k^*(t) = f_k(\mathbf{h}(t)), \quad \mathbf{h}(t) = \mathbf{o}_{t_i} \odot \tanh(c(t))$

Il Neural Hawkes supera i processi parametrici su task di predizione dei prossimi eventi in sequenze complesse (e-commerce, healthcare, finanza).

Glossario dei Simboli del Volume IV

A.1 Information Geometry

Simbolo	Significato
$\mathbf{G}(\boldsymbol{\theta})$	Fisher Information Matrix (FIM): $G_{ij} = \mathbb{E}_p[\partial_i \log p \cdot \partial_j \log p]$
$ds^2 = d\boldsymbol{\theta}^T \mathbf{G} d\boldsymbol{\theta}$	Metrica di Fisher-Rao (distanza infinitesima tra distribuzioni)
$\tilde{\nabla}\mathcal{L} = \mathbf{G}^{-1}\nabla\mathcal{L}$	Natural gradient
$\boldsymbol{\Omega}, \boldsymbol{\Gamma}$	Fattori Kronecker in K-FAC ($\boldsymbol{\Omega} = \mathbb{E}[\mathbf{g}\mathbf{g}^T]$, $\boldsymbol{\Gamma} = \mathbb{E}[\mathbf{a}\mathbf{a}^T]$)
$\chi_1 = \sigma_w^2 \mathbb{E}[\phi'(z)^2]$	Derivata della mappa di correlazione (mean field) — determina il regime
q^l (mean field)	Varianza delle pre-activazioni al layer l
c^l (mean field)	Correlazione tra due input al layer l
$\text{grad}_{\mathcal{M}} f = \Pi_{T_{\mathbf{x}}\mathcal{M}}(\nabla f)$	Riemannian gradient (proiezione sullo spazio tangente)
$R_{\mathbf{x}}$	Retrazione dalla varietà $\mathcal{M}$
$\hat{\boldsymbol{\theta}} = \mathbf{X}^+\mathbf{y}$	Soluzione a norma minima (pseudo-inversa — implicit bias GD)
$\hat{\mathbf{W}}$	Max-margin classifier (implicit bias GD per classificatori)

A.2 Interpretabilità Meccanicistica

Simbolo	Significato
$h_n^l(\mathbf{x})$	Attivazione del neurone n al layer l su input $\mathbf{x}$
$\mathbf{W}_{\text{enc}}, \mathbf{W}_{\text{dec}}$	Encoder e decoder dello Sparse Autoencoder
$\mathbf{f} \in \mathbb{R}^M$	Feature sparse dell'SAE ($M \gg d$)
$\text{DE}(h)$	Direct effect dell'attenzione head h sul logit target
$\mathbf{x}^l$	Residual stream al layer l del Transformer
$\mathbf{W}_U$	Unembedding matrix (da residual stream a logit)
$\mathbf{T}_l$	Trasformazione affine del Tuned Lens al layer l
$\Lambda(x)$	Likelihood ratio dell'attacco LiRA (membership inference)

A.3 Causalità e Scaling

Simbolo	Significato
$\text{Pa}(V_i)$	Genitori causali della variabile V_i nel DAG
$do(X = x)$	Operatore di intervento (rimuove tutte le cause di X)
$p(Y \mid do(X = x))$	Distribuzione interventistica di Y sotto $do(X = x)$
$D_\alpha(P\|Q)$	Rényi divergence di ordine α
$L(N, D) = E + A/N^\alpha + B/D^\beta$	Legge di scaling Chinchilla
$N^* \approx 0.5\sqrt{C/6}$	Numero parametri ottimale (Chinchilla)
$b' = b(L/L_{\text{train}})^{d/(d-2)}$	Base RoPE scalata per context extension

A.4 Privacy Differenziale e Audio

Simbolo	Significato
(ε, δ)-DP	Differential privacy con parametri ε (privacy loss) e δ (failure prob.)
$\Delta_2 f$	L2 global sensitivity della query f
σ (noise multiplier)	Std del rumore gaussiano in DP-SGD $= \sigma \cdot C/B$
C (clipping threshold)	Soglia di clipping del gradiente per-sample in DP-SGD
$q = B/N$ (sampling rate)	Frazione del dataset per mini-batch
(α, ε)-RDP	Rényi Differential Privacy di ordine α
$X[m, k]$	STFT: coefficiente al frame m e frequenza k
$S[m, t]$ (mel)	Mel spectrogram al bin m e frame t
$\alpha_t(u)$ (CTC)	Forward variable CTC: probabilità di emettere $\mathbf{y}_{1:u}$ entro il frame t
$\varnothing$ (blank)	Token di blank CTC
$(\mathbf{f}_t, \mathbf{g}_u)$	Acoustic encoding e prediction network output (RNN-T)

A.5 Robotica e Biologia

Simbolo	Significato
$\boldsymbol{\epsilon}_\theta(\mathbf{a}^k, k, \mathbf{o})$	Denoising network per diffusion policy
$(T_i, \mathbf{t}_i) = (R_i, \mathbf{t}_i)$	Frame rigido del residuo i (AlphaFold2)
$\mathbf{m} \in \mathbb{R}^{S \times L \times c_m}$	MSA representation (AlphaFold2)
$\mathbf{z} \in \mathbb{R}^{L \times L \times c_z}$	Pair representation (AlphaFold2)
$w^{(k)} = e^{-S(\tau^{(k)})/\lambda}$	Pesi MPPI per importance sampling
$W(d_{ij})$	Continuous filter di distanza (SchNet)
$e_k(d) = \exp(-\gamma(d - \mu_k)^2)$	Radial Basis Function per distanze (SchNet)
$\mathbf{V}_i \in \mathbb{R}^{F \times 3}$	Feature vettoriali equivarianti (PaiNN)
$\hat{\mathbf{r}}_{ij} = (\mathbf{r}_j - \mathbf{r}_i)/d_{ij}$	Vettore unitario interatomico

Identità Matematiche del Volume IV

B.1 Information Geometry e Ottimizzazione

Fisher e Natural Gradient	
$G_{ij}(\boldsymbol{\theta}) = \mathbb{E}_p[\partial_i \log p \cdot \partial_j \log p] = -\mathbb{E}_p[\partial_i \partial_j \log p]$	FIM — due forme equivalenti
$D_{\mathrm{KL}}(p(\cdot;\boldsymbol{\theta}) \Vert p(\cdot;\boldsymbol{\theta}+d\boldsymbol{\theta})) = \frac{1}{2} d\boldsymbol{\theta}^T \mathbf{G} d\boldsymbol{\theta} + O(\vert d\boldsymbol{\theta}\vert^3)$	KL $\approx$ metrica quadratica al 2° ordine
$\tilde{\nabla}\mathcal{L} = \mathbf{G}^{-1}\nabla\mathcal{L}$	Natural gradient — invariante a reparametrizzazione
$\Delta\mathbf{W} = -\eta\boldsymbol{\Omega}^{-1}\nabla_{\mathbf{W}}\mathcal{L}\boldsymbol{\Gamma}^{-1}$	K-FAC update
$(\mathbf{A}\otimes\mathbf{B})^{-1} = \mathbf{A}^{-1}\otimes\mathbf{B}^{-1}$	Inversione prodotto Kronecker
Mean Field Theory	
$q^l = \sigma_w^2 \mathbb{E}_{z\sim\mathcal{N}(0,q^{l-1})}[\phi(z)^2] + \sigma_b^2$	Propagazione della varianza (mean field)
$\chi_1 = \sigma_w^2 \mathbb{E}[\phi'(z)^2]$ con $\chi_1 \gtrless 1 \to$ ordine/caos	Transizione di fase ordine-caos
$\chi_1^{\mathrm{ReLU}} = \sigma_w^2/2 = 1 \Rightarrow \sigma_w^2 = 2/N$	Inizializzazione He = edge of chaos
Implicit Bias	
GD da $\theta_0 = 0 \to \hat{\theta} = \mathbf{X}^+\mathbf{y}$ (norma minima)	Regressione lineare
GD per classif. lineare $\to \hat{\theta}/\Vert\hat{\theta}\Vert \to \hat{\theta}_{\mathrm{SVM}}$	Max-margin (loss logistica)
GD reti profonde $\to \Vert\mathbf{W}\Vert_*$ minima	Norma nucleare = rango basso

B.2 Causalità e Privacy

Do-Calculus	
$p(Y\vert do(X=x)) = \sum_z p(Y\vert X=x, Z=z)p(Z=z)$	Backdoor adjustment formula
$p(Y\vert do(X)) = \sum_m \sum_{x'} p(M\vert X)p(Y\vert M, X=x')p(X=x')$	Frontdoor adjustment formula
Differential Privacy	
$\mathbb{P}[\mathcal{M}(D)\in S] \le e^{\varepsilon}\mathbb{P}[\mathcal{M}(D')\in S] + \delta$	(ε,δ)-DP definition
$\sigma_{\mathrm{GM}} = \frac{\Delta_2 f\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$	Noise calibration meccanismo gaussiano
k meccanismi (ε,δ)-DP $\to (O(\varepsilon\sqrt{k}), k\delta+\delta')$-DP	Advanced Composition
(α,ε)-RDP $\times k \to (\alpha, k\varepsilon)$-RDP	RDP composition lineare
(α,ε)-RDP $\to (\varepsilon + \frac{\log(1/\delta)}{\alpha-1}, \delta)$-DP	RDP $\to$ DP conversion

B.3 Audio, Biologia e Robotica

Audio e Speech	
$X[m,k] = \sum_n x[mH+n]w[n]e^{-j2\pi kn/N}$	STFT
$m = 2595\log_{10}(1+f/700)$	Conversione Hz $\to$ mel
$p(\mathbf{y}\|\mathbf{x}) = \sum_{\boldsymbol{\pi}\in\mathcal{B}^{-1}(\mathbf{y})}\prod_t p(\pi_t\|\mathbf{x})$	CTC likelihood
$\alpha_t(u) \leftarrow [\alpha_{t-1}(u)+\alpha_{t-1}(u-1)]\cdot p(y_u\|\mathbf{x}_t)+\alpha_{t-1}(u)p(\varnothing\|\mathbf{x}_t)$	CTC forward algorithm
AlphaFold2 e Molecole	
$\mathbf{z}_{ij} \leftarrow \mathbf{z}_{ij} + \mathrm{Lin}(\frac{1}{S}\sum_s \mathbf{m}_{si}\otimes\mathbf{m}_{sj})$	Outer product mean (Evoformer)
$a_{ij} \propto q_i^T k_j - \frac{\gamma}{2}\|T_i q_i^p - T_j k_j^p\|^2$	Invariant Point Attention (IPA)
$\mathbf{v}_i = \sum_{j\in\mathcal{N}(i)} \mathbf{h}_j \odot W(d_{ij})$	SchNet interaction
$e_k(d) = \exp(-\gamma(d-\mu_k)^2)$	SchNet radial basis
MPPI e Diffusion Policy	
$w^{(k)} = e^{-S(\tau^{(k)})/\lambda}$, $\bar{\mathbf{a}} \leftarrow \frac{\sum w^{(k)}(\bar{\mathbf{a}}+\boldsymbol{\epsilon}^{(k)})}{\sum w^{(k)}}$	MPPI update
$\mathrm{FiLM}(\mathbf{h},\mathbf{o}) = \gamma(\mathbf{o})\odot\mathbf{h}+\beta(\mathbf{o})$	Feature-wise Linear Modulation

Diagnostica e Iperparametri del Volume IV

C.1 Diagnostica

Contesto	Sintomo	Causa
Natural Gradient	K-FAC instabile o diverge	FIM mal condizionata, σ_w sbagliato
Mean Field	Gradiente esplode/svanisce	Regime caotico/ordinato
IRM	Non converge a soluzione invariante	Ambienti non sufficientemente diversi
DP-SGD	Modello non converge	Noise multiplier σ troppo alto
CTC	Training instabile nei primi step	Log-zero su sequenze brevi
Diffusion Policy	Azioni disomogenee/jerky	Horizon H troppo corto o T_o insufficiente
AlphaFold2-like	TM-score basso su lunghe proteine	Domain drop durante MSA, alta entropia evo
MusicGen	Audio incoerente dopo pochi secondi	Context length superato per token EnCodec

C.2 Iperparametri del Volume IV

Metodo	Iperparametro	Tipico	Note
K-FAC	Damping λ	10^{-3}–10^{-1}	Aumentare se ir
Update FIM ogni	10–100 step	Più frequente = più preciso ma costoso	
Update inverso ogni	200–1000 step	Molto costoso, ammortizzare	
DP-SGD	Clipping C	1.0	Stimare come m
Noise multiplier σ	1.1–1.5	Più alto = più privacy, meno utility	
Target ε	1.0–10.0	$\varepsilon < 1$: forte, $\varepsilon > 10$: debole	
CTC/RNN-T	Beam width	5–20	Più largo = mig
Language model weight	0.3–0.8	Peso del LM nel shallow fusion	
Diffusion Policy	Denoising steps K	100	DDPM; DDIM
Horizon H (azioni)	16	Frame di azione futura	
Observation horizon T_o	2	Frame di osservazione passata	
SchNet/DimeNet	Cutoff $d_{\max}$	5–10 Å	Oltre questa dis
RBF gaussiane K	300	Per la funzione di filtro	

Complessità Computazionale del Volume IV

Algoritmo	Compute	Memoria	Note
Natural Gradient (esatto)	$O(\|\theta\|^3)$	$O(\|\theta\|^2)$	Inapplicabile a reti grandi
K-FAC per layer $(d_{\text{in}}, d_{\text{out}})$	$O(d_{\text{in}}^3 + d_{\text{out}}^3)$	$O(d_{\text{in}}^2 + d_{\text{out}}^2)$	Ammortizzato su T_{inv} step
DP-SGD per-sample gradients	$O(B \cdot \|\theta\|)$	$O(B \cdot \|\theta\|)$	$B\times$ più lento di SGD standa
CTC forward-backward	$O(T \cdot U)$	$O(T \cdot U)$	T=frame audio, U=lunghezz
RNN-T per sequenza	$O(T \cdot U \cdot d)$	$O(T \cdot U)$	Grafo di allineamento compl
Evoformer (AlphaFold2)	$O(S^2 L + S L^2)$	$O(SL \cdot c_m + L^2 \cdot c_z)$	S=sequenze MSA, L=lungh
IPA (structure module)	$O(L^2)$	$O(L^2)$	Come full attention su residu
MPPI (K rollouts, H orizzonte)	$O(K \cdot H \cdot d_{\text{model}})$	$O(K \cdot H)$	Parallelo su GPU
SchNet per molecola (N atomi)	$O(N^2)$	$O(N)$	Con cutoff: $O(N \cdot k)$ dove k
RVQ encoding (EnCodec)	$O(T \cdot Q \cdot K)$	$O(Q \cdot K)$	Q=quantizer, K=codebook

Bibliografia Completa del Volume IV

E.1 Information Geometry e Ottimizzazione

- Amari, S. (1998). *Natural Gradient Works Efficiently in Learning.* Neural Computation 10(2).

- Martens, J., Grosse, R. (2015). *Optimizing Neural Networks with Kronecker-factored Approximate Curvature (K-FAC).* ICML 2015.
- Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., Ganguli, S. (2016). *Exponential Expressivity in Deep Neural Networks through Transient Chaos.* NeurIPS 2016.
- Schoenholz, S.S., Gilmer, J., Ganguli, S., Sohl-Dickstein, J. (2017). *Deep Information Propagation.* ICLR 2017.
- Soudry, D., Hoffer, E., Nacson, M.S., Gunasekar, S., Srebro, N. (2018). *The Implicit Bias of Gradient Descent on Separable Data.* JMLR 19(70).
- Gunasekar, S., Woodworth, B., Bhojanapalli, S., Neyshabur, B., Srebro, N. (2017). *Implicit Regularization in Matrix Factorization.* NeurIPS 2017.
- Frankle, J., Dziugaite, G.K., Roy, D.M., Carlin, M. (2020). *Linear Mode Connectivity and the Lottery Ticket Hypothesis.* ICML 2020.
- Ainsworth, S.K., Hayase, J., Srinivasa, S. (2023). *Git Re-Basin: Merging Models modulo Permutation Symmetries.* ICLR 2023.

E.2 Interpretabilità e Causalità

- Mahendran, A., Vedaldi, A. (2015). *Understanding Deep Image Representations by Inverting Them.* CVPR 2015.
- Elhage, N., et al. (2022). *Toy Models of Superposition.* Transformer Circuits Thread.
- Bricken, T., Templeton, A., et al. (2023). *Towards Monosemanticity: Decomposing Language Models with Dictionary Learning.* Anthropic Technical Report.
- Elhage, N., et al. (2021). *A Mathematical Framework for Transformer Circuits.* Transformer Circuits Thread.
- Meng, K., Bau, D., Andonian, A., Belinkov, Y. (2022). *Locating and Editing Factual Associations in GPT (ROME).* NeurIPS 2022.
- Belrose, N., Furman, Z., Smith, L., Halawi, D., Oikarinen, T., Du, Y., Bhatt, U., Steinhardt, J. (2023). *Eliciting Latent Predictions from Transformers with the Tuned Lens.* arXiv:2303.08112.
- Pearl, J. (2000). *Causality: Models, Reasoning, and Inference.* Cambridge University Press.

- Arjovsky, M., Bottou, L., Gulrajani, I., Lopez-Paz, D. (2019). *Invariant Risk Minimization.* arXiv:1907.02893.
- Yang, F., Yang, Z., Cohen, W.W. (2017). *Differentiable Learning of Logical Rules for Knowledge Base Reasoning (Neural-LP).* NeurIPS 2017.

E.3 Scaling Laws e Privacy

- Kaplan, J., et al. (2020). *Scaling Laws for Neural Language Models.* arXiv:2001.08361.
- Hoffmann, J., et al. (2022). *Training Compute-Optimal Large Language Models (Chinchilla).* NeurIPS 2022.
- Wei, J., et al. (2022). *Emergent Abilities of Large Language Models.* TMLR 2022.
- Schaeffer, R., Miranda, B., Koyejo, S. (2023). *Are Emergent Abilities of Large Language Models a Mirage?* NeurIPS 2023.
- Kwon, W., et al. (2023). *Efficient Memory Management for Large Language Model Serving with PagedAttention (vLLM).* SOSP 2023.
- Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., Liu, Y. (2021). *RoFormer: Enhanced Transformer with Rotary Position Embedding (RoPE).* arXiv:2104.09864.
- Dwork, C., McSherry, F., Nissim, K., Smith, A. (2006). *Calibrating Noise to Sensitivity in Private Data Analysis.* TCC 2006.
- Abadi, M., et al. (2016). *Deep Learning with Differential Privacy (DP-SGD).* CCS 2016.
- Mironov, I. (2017). *Rényi Differential Privacy.* CSF 2017.
- Carlini, N., et al. (2022). *Membership Inference Attacks From First Principles (LiRA).* IEEE SP 2022.

E.4 Audio, Robotica e Biologia

- Graves, A., Fernández, S., Gomez, F., Schmidhuber, J. (2006). *Connectionist Temporal Classification (CTC).* ICML 2006.
- Graves, A. (2012). *Sequence Transduction with Recurrent Neural Networks (RNN-T).* ICML Workshop 2012.
- Kong, J., Kim, J., Bae, J. (2020). *HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis.* NeurIPS 2020.

- Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I. (2022). *Robust Speech Recognition via Large-Scale Weak Supervision (Whisper).* ICML 2023.
- Kim, J., Kong, J., Son, J. (2021). *Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech (VITS).* ICML 2021.
- Copet, J., et al. (2023). *Simple and Controllable Music Generation (MusicGen).* NeurIPS 2023.
- Défossez, A., Copet, J., Synnaeve, G., Adi, Y. (2022). *High Fidelity Neural Audio Compression (EnCodec).* TMLR 2023.
- Chi, C., Xu, Z., Feng, S., et al. (2023). *Diffusion Policy: Visuomotor Policy Learning via Action Diffusion.* RSS 2023.
- Brohan, A., et al. (2022). *RT-1: Robotics Transformer for Real-World Control at Scale.* RSS 2023.
- Brohan, A., et al. (2023). *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control.* CoRL 2023.
- Williams, G., Wagener, N., Goldfain, B., et al. (2017). *Information Theoretic MPC for Model-Based Reinforcement Learning (MPPI).* ICRA 2017.
- Jumper, J., et al. (2021). *Highly Accurate Protein Structure Prediction with AlphaFold.* Nature 596.
- Lin, Z., et al. (2022). *Language Models of Protein Sequences at the Scale of Evolution Enable Accurate Structure Prediction (ESMFold).* Science 379.
- Schütt, K.T., Kindermans, P.J., Sauceda, H.E., Chmiela, S., Tkatchenko, A., Müller, K.R. (2017). *SchNet: A Continuous-filter Convolutional Neural Network for Modeling Quantum Interactions.* NeurIPS 2017.
- Gasteiger, J., Groß, J., Günnemann, S. (2020). *Directional Message Passing for Molecular Graphs (DimeNet).* ICLR 2020.
- Schütt, K.T., Unke, O.T., Gastegger, M. (2021). *Equivariant Message Passing for the Prediction of Tensorial Properties and Molecular Spectra (PaiNN).* ICML 2021.
- Batzner, S., et al. (2022). *E(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials (NequIP).* Nature Communications.

- Avsec, Z., et al. (2021). *Effective Gene Expression Prediction from Sequence by Integrating Long-Range Interactions (Enformer).* Nature Methods.